

# Computation Support in DROP

**v4.99** 8 August 2020

**Kubernetes**

**Overview**

1. Focus of the Kubernetes System: *Kubernetes* is an open-source container orchestration system for automating computer application deployment, scaling, and management (Garrison (2016), Wikipedia (2020)).
2. Objectives of the System Design: It aims to provide a platform for automating deployment, scaling, and operations of application containers across clusters of hosts. It works with a range of container tools, including Docker.
3. Vendor Service Deployments using Kubernetes: Many closed services offer a Kubernetes-based platform or infrastructure as a service (PaaS or IaaS) on which Kubernetes can be deployed as a platform providing service. Many vendors also provide their own branded Kubernetes distributions.

**Kubernetes Objects**

1. Purpose behind the Kubernetes Objects: Kubernetes maintains a set of building blocks or primitives which collectively provide mechanisms that deploy, maintain, and scale applications based on CPU, memory, or custom metrics (Sharma (2017)).
2. Structure and Extensibility of Kubernetes: Kubernetes is loosely coupled and extensible to meet different workloads. The extensibility is provided in large part by the Kubernetes API, which is used by internal components as well as extensions and containers that run on Kubernetes. The platform exerts its control over compute and storage resources by defining resources as Objects, which can then be managed as such.

**Pods**

1. What is a Kubernetes Pod? A *pod* is a higher level of abstraction grouping containerized components. A pod consists of one or more containers that are guaranteed to be co-located on the host machine and can share resources. The basic scheduling unit in Kubernetes is a *pod*.
2. IP Addresses for Kubernetes Pods: Each pod in Kubernetes is assigned a unique *Pod IP address* within the cluster, which allows applications to use ports without the risk of conflict (Langemak (2015a)). Within the pod, all containers can reference each other on localhost, but a container within one pod has no way of directly addressing another container within another pod; for that, it has to use the pod IP address.
3. Name Service to Reference Pods: An application developer should never use the pod IP address though, to reference/invoke a capability in another pod, as pod IP addresses are ephemeral – the specific pod that they are addressing may be assigned to another pod IP address on re-start. Instead, they should use a reference to a Service, which holds a reference to the target pod at the specific pod IP address.
4. Local/Network Disk Volume Pods: A pod can define a Volume, such as a local disk directory or a network disk, and expose it to the containers in the pod (Strachan (2015a)). Pods can be managed manually through the Kubernetes API, or their management can be delegated to a controller.
5. Storage for ConfigMaps and Secrets: Such volumes are also the basis for Kubernetes features of ConfigMaps – which provide access to configuration through the file-system visible to the container – and Secrets – which provide access to credentials required to access remote resources securely by providing those credentials on the filesystem visible only to authorized containers.

**ReplicaSets**

1. Purpose of the ReplicaSet: A ReplicaSet’s purpose is to maintain a stable set of replica pods running at any given time. As such, it is often used to guarantee the availability of a specified number of identical pods.
2. Selector Based ReplicaSet Specification: The ReplicaSets can also be said to be a grouping mechanism that lets Kubernetes maintain the number of instances that have been declared for a given pod. The definition of ReplicaSet uses a selector whose evaluation will result in identifying all pods that are associated with it.

**Services**

1. Definition of a Kubernetes Service: A Kubernetes source is a set of pods that work together, such as one-tier of a multi-tier application. The set of pods that constitute a service is defined by a label selector.
2. Modes of Service Discovery in Kubernetes: Kubernetes provides two modes of service discovery, using environmental variables of Kubernetes DNS. Service discovery assigns a stable IP address and DNS name to the service, and load balances traffic in a round-robin manner to network connections of that IP address among the pods matching the selector – even as failures cause the pods to move from machine to machine.
3. Exposing Front/Back End Services: By default, a service is exposed inside a cluster, e.g., back end pods might be grouped into a service, with requests from the front-end pods load balanced among them, but a service can also be exposed outside a cluster, e.g., for clients to reach front-end pods (Langemak (2015b)).

**Volumes**

1. Transient Nature of Kubernetes Filesystems: Filesystems in Kubernetes containers provide ephemeral storage, by default. This means that re-start of the pod will wipe out any data on such containers, and therefore, this form of storage is quite limiting in anything but trivial applications.
2. Persistent Storage Mechanism - Kubernetes Volume: A Kubernetes Volume provides persistent storage that exists for the life of the pod itself. This storage can also be used as shared disk space for containers within the pod.
3. Restrictions on the Mount Points: Volumes are mounted at specific mount points within the container, which are defined by the pod configuration, and cannot mount onto other Volumes or link to other Volumes. The same Volume can be mounted at different points in the filesystem tree by different containers.

**Namespaces**

Kubernetes provides a partitioning of the resources it manages into non-overlapping sets called namespaces. They are intended for use in environments with many users spread across multiple teams or projects, even separating environments like development, test, and production.

**ConfigMaps and Secrets**

1. Types of Kubernetes Configuration Data: A common application challenge is deciding where to store and manage configuration information, some of which may contain sensitive data. Configuration data can be anything as fine-grained as individual properties or coarse-grained information like entire configuration files or JSON/XML documents.
2. Access to Configuration Data: Kubernetes provides two closely related mechanisms to deal with this need – *configmaps* and *secrets*, both of which allow for configuration changes to be made without requiring an application rebuild. The data from configmaps and secrets will be made available to every single application to which these objects have been bound via deployment.

**StatefulSets**

1. Kubernetes Stateless versus Stateful Scaling: It is very easy to address the scaling of stateless applications: one simply adds more running pods – which is something that Kubernetes does very well. Stateful workloads are much harder, because the state needs to be preserved if a pod is restarted, and if the application is scaled up or down the state may need to be redistributed.
2. Ordering of the Stateful Instances: Databases are an example of stateful workloads. When run in high-availability mode, many databases come with the notion of a primary instance and secondary instance(s). In this case, the notion of ordering the instances is important.
3. Uniqueness of the Stateful Instances: Other applications like Kafka distribute the data among their brokers – so that one broker is not the same as another. In this case, the notion of instance uniqueness is important.
4. Uniqueness and Ordering among Pod Instances: StatefulSets are controllers that are provided by Kubernetes that enforce the properties of uniqueness and ordering among the instances of a pod and can be used to run stateful applications.

**DaemonSets**

Normally, the location where the pods are run are determined by the algorithm implemented in the Kubernetes scheduler. The ability to do this kind of pod scheduling is implemented by the feature called DaemonSets.

**Secrets**

Secrets contain ssh keys, passwords, and OAuth tokens for the pod.

**Managing Kubernetes Objects: Labels and Selectors**

1. Label Selector Based Object Matches: Kubernetes enables clients – users or internal components – to attach keys called *labels* to any API Object in the system, such as pods or nodes. Correspondingly, *label selectors* are queries against labels that resolve to matching objects.
2. Service Routing using Label Selectors: When a service is defined, one can define the label selector that will be used by the service router/load balancer to select the pod instances that the traffic will be routed to.
3. Controlling Service Routing Using Labels: Thus, simply changing the labels of the pods or changing the label selectors on the service can be used to control which pod gets the traffic and which doesn’t, and this can be used to support various deployment patterns like blue-green deployments or A-B testing. This capability to dynamically control how services utilize implementing resources provides a loose coupling within the infrastructure.
4. Multi-Criterion Label Selection Example: For example, if the application’s pods have labels for a system – with values such as , , for example – and a – with values such as , , for example – then an operation on all of and nodes can use a label selector, such as:

**Field Selectors**

1. Object-Attribute-Value Based Selection: Just like fields, field selectors also let one select Kubernetes resources. Unlike labels, the selection is based on the attribute values inherent to the resource being selected, rather than user-defined categorization.
2. Built-in Metadata Field Selectors: and are field selectors that will be present on all Kubernetes objects. Other selectors can be used depending on the object/resource type.

Replication Controllers and Deployments

**References**

* Garrison, J. (2016): [Why Kubernetes is Abbreviated k8s](https://medium.com/@rothgar/why-kubernetes-is-abbreviated-k8s-905289405a3c)
* Langemak, J. (2015a): [Kubernetes 101 – Networking](http://www.dasblinkenlichten.com/kubernetes-101-networking/)
* Langemak, J. (2015b): [Kubernetes 101 – External Access into the Cluster](http://www.dasblinkenlichten.com/kubernetes-101-external-access-into-the-cluster/)
* Sharma, P. (2017): [Autoscaling based on CPU/Memory in Kubernetes — Part II](https://blog.powerupcloud.com/autoscaling-based-on-cpu-memory-in-kubernetes-part-ii-fe2e495bddd4)
* Strachan, J. (2015): [Kubernetes for Developers](https://blog.fabric8.io/kubernetes-for-developers-2a9c7202fcd3#.b6u76jxar)
* Wikipedia (2020): [Kubernetes](https://en.wikipedia.org/wiki/Kubernetes)