
# Kubernetes, UberCloud Containers, and HPC

UberCloud Whitepaper by Daniel Gruber, Burak Yenier, and Wolfgang Gentzsch
September 17, 2019

Software containers and Kubernetes are important tools for building, deploying, running, and managing modern enterprise applications at scale and delivering enterprise software faster and more reliably to the end user - while using resources more efficiently and reducing costs. Recently, High Performance Computing (HPC) is moving closer to the enterprise and can therefore benefit from a container and Kubernetes ecosystem, with new requirements to quickly allocate and deallocate computational resources to HPC workloads. Compute capacity, be it for enterprise or HPC workload, can no longer be planned years in advance.

Getting on-demand resources from a shared compute resource pool has never been easier as cloud service providers and software vendors are continuously investing and improving their services. In enterprise computing, packaging software in container images and running containers has been adopted quickly and is now a standard approach. Kubernetes has become the most widely used container and resource orchestrator in the Fortune 500 companies. The HPC community, led by efforts in the AI community, is picking up the concept and applying it to batch jobs and interactive applications.

While containers are simplifying software packaging, delivery, and execution in a repeatable and portable way, Kubernetes provides capabilities to orchestrate a multitude of them. Kubernetes simplifies container deployment and management by providing a common high abstraction layer, both on-premises and at all major cloud providers. This abstraction allows system administrators to quickly deploy, scale, and remove applications and computational workloads in dedicated or shared environments virtually everywhere - in-house and in dynamic cloud environments.

Cloud environments provide engineers the benefit of quicker access to faster machines, newer GPUs, and arbitrary sized compute pools for batch processing in a self-service way. For IT organizations, which use Kubernetes for enterprise computing, adding engineering applications would mean that they can fully apply their existing experience with Kubernetes and tooling across different workloads, now including HPC. And, by exploiting a shared pool of resources under-utilization can be minimized.

UberCloud HPC containers fully embrace the Kubernetes container orchestrator. This article provides you with key insights beneficial for all executives dealing with engineering/HPC workloads in your organization. For IT leaders who already are running or testing Kubernetes this article provides guidance about how engineers in your organization will benefit from a central container platform.


# Containers and HPC: Why Should We Care?

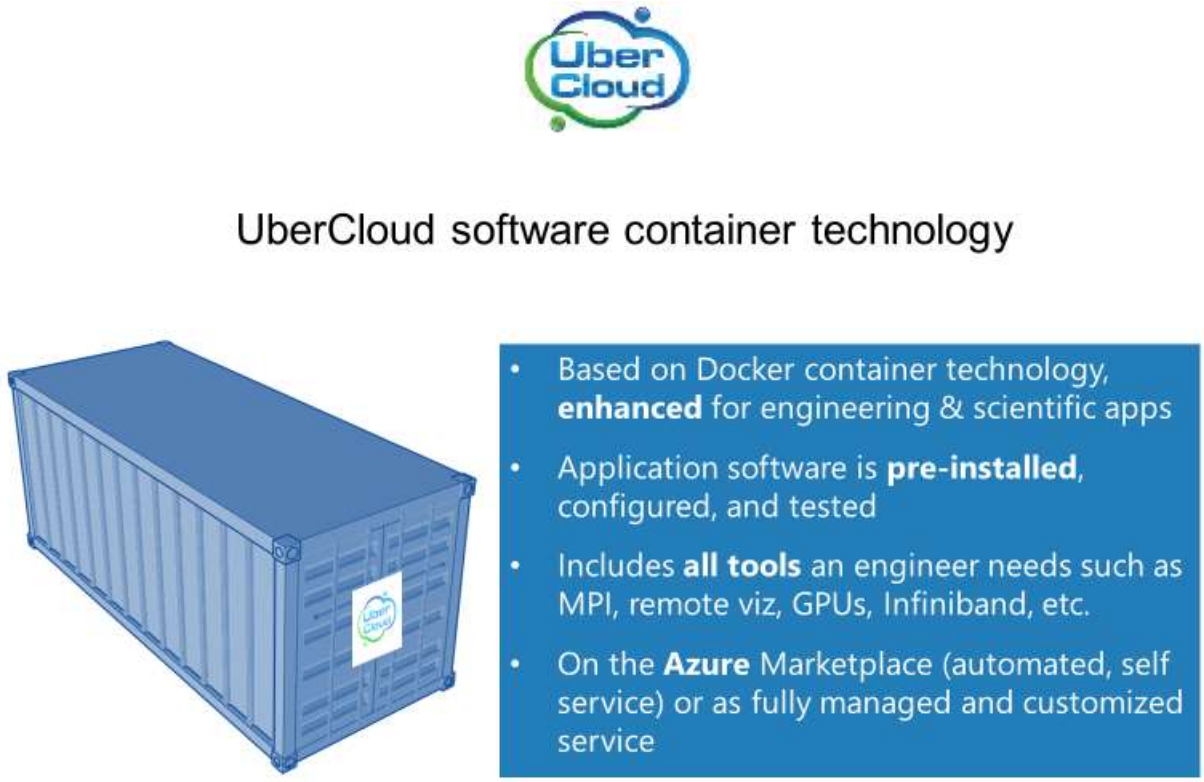
HPC leaders have a hard time. There are lots of changes and new ways of thinking in software technology and IT operations. Containerization have become ubiquitous, container orchestration with Kubernetes gets deployed now in most of the data centers of the Fortune 500 companies, Deep Learning workloads are continuously increasing their footprint, and Site Reliability Engineering (SRE) has been adopted on many sites.

But in HPC data centers there are still lots of workloads running in the same way they did decades ago. There is a very good reason for this stagnation. Established HPC technologies often outperform newly appearing solutions in enterprise computing. It is very hard for each new technology to judge its usefulness for HPC type of workloads. In general, there are always tradeoffs between new and established technologies.

In this article we share our experiences on running HPC workloads in bare containers as well as orchestrating them in a higher level on Kubernetes.

## UberCloud and Containers

UberCloud has been building containers for HPC users since the first beta release of Docker in early 2013. When Docker was accessible for the first time UberCloud was one of the first companies embracing this new technology for simplifying deployment of HPC applications in cloud environments for UberCloud customers. In the following years UberCloud containers got more functionalities, like an integrated HPC workload manager, resource monitoring, MPI, InfiniBand, GPU support, and more, for engineering applications like ANSYS, COMSOL, STAR-CCM+, and OpenFOAM, and recently also for HPC-intensive AI applications using Deep Learning. Delivering those applications in containers comes with huge benefits, but also challenges. Containers make it easy to deploy applications in different environments since dependency management is largely solved. To make deployment of UberCloud containers even easier, UberCloud also integrated them into specific solutions like Azure CycleCloud and letting them run under the control of Univa Grid Engine. In the last two years, we also integrated them with datacenter tools like DockerEE, VMware's vSphere, OpenStack, Red Hat OCP, and HPE's Hybrid HPC Cloud Infrastructure.




## Introduction to Kubernetes

If your engineers or operators are running a single container on their laptop they probably use Docker for doing that. But when having multiples of containers potentially on dozens or hundreds of machines it becomes difficult to get them maintained. Kubernetes simplifies container orchestration by providing scheduling, container life-cycle management, networking functionalities and more in a scalable and extensible platform.

Major components of Kubernetes are the Kubernetes master which contains an API server, scheduler, and a controller manager. Controllers are a main concept: they watch out for the current state of resources and compare them with the expected state. If they differ, they take actions to move to an expected state. On the execution side we have the kubelet which is in contact with the master as well as a network proxy. The kubelet manages containers by using the container runtime interface (CRI). Which container runtime (like Docker, containerd, CRI-O) is finally executing the container is configurable.

## Running Kubernetes or HPC Schedulers?

Kubernetes is doing workload and resource management. Sounds familiar? Yes, in many ways it shares lots of functionalities with traditional HPC workload managers. The main differences are


the workload types they focus on. While HPC workload managers are focused on running distributed memory jobs and support high-throughput scenarios, Kubernetes is primarily built for orchestrating containerized microservice applications.

HPC workload managers like Univa Grid Engine added a huge number of features in the last decades. Some notable functionalities are:

- Support for shared and distributed memory (like MPI based) jobs
- Advance reservations for allocating and blocking resources in advance
- Fair-share to customize resource usage patterns across users, projects, and departments
- Resource reservation for collecting resources for large jobs
- Preemption for stopping low prior jobs in favor for running high prior jobs
- NUMA aware scheduling for automatically allocating cores and sockets for faster executing jobs
- Adhere to standards for job submission and management (like DRMAA)

HPC workload managers are tuned for speed, throughput, and scalability, being capable of running millions of batch jobs a day and supporting the infrastructure of the largest supercomputers in the world. What traditional HPC workload managers lack are means for supporting microservice architectures, deeply integrated container management capabilities, network management, and application life-cycle management. They are primarily built for running batch jobs in different scenarios like high-throughput, MPI jobs spanning across potentially hundreds or thousands of nodes, jobs running weeks, or jobs using special resource types (GPUs, FPGAs, licenses, etc.).

Kubernetes on the other hand is built for containerized microservice applications from the bottom-up. Some notable features are:

- Management of sets of pods. Pods consist of one or more co-located containers.
- Networking functionalities through a pluggable overlay network
- Self-healing through a controller concept which compares an expected state with the current state and takes actions to align both
- Declarative style configuration
- Load balancing functionalities
- Rolling updates of different versions of workloads
- Integrations in many monitoring and logging solutions
- Hooks to integrate external persistent storage in pods
- Service discovery and routing

What Kubernetes lacks at this time is a proper high-throughput batch job queueing system with a sophisticated rule system for managing resource allocations. But one of the main drawbacks we see is that traditional HPC engineering applications are not yet built to interact with Kubernetes. But this will change in the future. New kinds of AI workloads on the other hand are supporting Kubernetes already very well - in fact many of these packages are targeted to Kubernetes.


# How and Why UberCloud Supports Kubernetes

Kubernetes is one of UberCloud's new target platforms for running UberCloud Containers. UberCloud Containers are built in a way that they are infrastructure and even workload orchestration independent. Why do we support Kubernetes and what does that mean?

At UberCloud we are combining both systems to meet the demanding requirements of our customers. Kubernetes is used for managing UberCloud containers along with all the helper services. Inside the containers we are running not just the engineering application, we also have the capability to either plug into an existing or run an entire HPC resource manager installation (like SLURM or Univa Grid Engine). In that way we provide compatibility to the engineering applications and can exploit the extended batch scheduling capabilities. At the same time our whole deployment can be operated in all Kubernetes enabled environments with the advantages of standardized container orchestration.

## Ease of Administration

HPC environments consist of a potentially large set of containers. Kubernetes helps to simplify deployments which results in faster start time of the HPC application running inside UberCloud's HPC containers. Overall the higher abstraction of administration compared to container runtime engines provides us the necessary flexibility we need to fulfill different customer requirements.

Management operations like scaling the deployment are much simpler to implement and execute. In that way we can provide a flexible self-service platform for the engineer which can deploy, scale, and destroy the HPC application rapidly. Our solution also fully supports CI/CD pipelines so that customers can fully integrate the deployment of HPC applications in their own solution.

The UberCloud operations team manages the infrastructure and the containers running on top. With Kubernetes, and Kubernetes as a Service solutions (e.g. Google Kubernetes Engine, GKE) our operations team as well as our customers, can define entire clusters and surrounding services as a Kubernetes deployment and let GKE manage individual containers running on individual hosts. Ease of administration is a critical component to reducing the administration burden and transferring the simplified day-to-day administration to our customers.


## The Run-Time for Hybrid and Multi-Cloud Offers True Portability

Portability is a key value of containers. We are able to run UberCloud's HPC Containers on all the different platforms our customers are using. At the same time, we are providing the same experience of our solution for on-premises as well as different cloud providers resources.

When we started with Docker container run time, we knew that it is going to be widely distributed and accessible by our customers. Due to the OCI compliance of our container images we are able to run HPC containers on any OCI compliant container platform available, hence also Kubernetes.

Kubernetes allows us to provide the same experience of our solution on-premises as well as on different cloud infrastructures. Our customers can seamlessly switch the infrastructure without any changes for the engineers. In that way they can choose the infrastructure by criteria like price, performance, and capabilities. When running on premises we can now offer a true hybrid-cloud experience by providing a consistent infrastructure with the same operational and HPC application experience and seamlessly use on-demand cloud resources when required.

# What's Next?

Embracing Kubernetes for the specific requirements of HPC and engineering workload is not straight forward. But due to the huge success of Kubernetes and its open and extensible architecture the ecosystem is opening up for HPC applications primarily driven by the demand of new AI workloads. UberCloud invests in making running engineering HPC workload in a shared on-premises or on-demand cloud environment as smooth as possible, offering and implementing supercomputing power as a service for the engineers.