Dina Lerjevik
Patricia Naccachian

Comparison of Kubernetes and Nomad Orchestration Tools: Finding Your Way Through the “Forest”

1. Introduction

There is an old proverb: “A single tree does not make a forest”. The same comes true for containers and orchestration tools, according to *Developer Relations expert* Matthew Revell[1]. Orchestration tools are essential when it comes to managing containers that are part of large IT-infrastructures[2,3]. However, there are plenty of options for orchestration tools. Hence, the aim of this essay is to provide assistance for navigating the “orchestration tool jungle”. The focus is to describe the architecture and scheduling of two commonly used tools, Kubernetes and Nomad. Moreover, a comparison will be made with regard to their architecture, abstraction level and integration with external tools.

1.1. What is a container?

In order to understand container orchestrators, we need to have some basic understanding of containers. Containers are software packages that include the libraries and dependencies needed to run an application reliably. Details such as configuration files, system tools and application binaries are included in the image of a container, which results in an execution environment which is self-contained[4]. This enables for migration between multiple ecosystems and for isolation of issues, which cuts down the amount of time spent debugging[5].

1.2. What is Container Orchestration?

Imagine a big tech company that makes use of containerization and that is looking to streamline their deployments using scaling or generate copies of their deployments in multiple environments. Performing these changes manually would likely turn into a challenging task[6,2]. For instance, microservices are containerized separately, meaning that a big enterprise might be dealing with thousands of containers[7]. Hence, there is a need for orchestration tools that automates the management of containers, including provisioning and deployment, scheduling, resource allocation, networking, balancing of workloads and scaling[6,4].

[1] Revell, Matthew. 2016. Introduction to container orchestration: Kubernetes, Docker Swarm and Mesos with Marathon. *Exoscale*. July 26. URL: https://www.exoscale.com/syslog/container-orchestration (Received: 2021-04-29)
[2] OpsWorks Co. 2020. A Look at the Best Container Orchestration Tools. January 7. URL: https://opsworks.co/container-orchestration-tools (Received: 2021-04-26)
[3] Eldridge, Isaac. 2018. What is Container Orchestration? July 17. *New Relic.* URL: https://newrelic.com/blog/best-practices/container-orchestration-explained (Received: 2021-04-30)
[4] Red Hat. n.d. What is Container Orchestration? URL: https://www.redhat.com/en/topics/containers/what-is-container-orchestration (Received: 2021-04-30)
[5] Gibb, Robert. 2019. What are containers? *StackPath*. May 30. URL: https://blog.stackpath.com/containers (Received: 2021-04-26)
[6] Conrad, John. 2020. What is Container Orchestration? *Capitalone*. August 24. URL: https://www.capitalone.com/tech/cloud/what-is-container-orchestration (Received: 2021-04-26)
[7] VMware. n.d. What is container orchestration? URL: https://www.vmware.com/topics/glossary/content/container-orchestration (Received: 2021-04-26)

1.3. Relevance for DevOps

DevOps software methodology highlights the importance of collaboration in order to speed up software production. However, working with DevOps is not equivalent to using a specific set of frameworks. One can implement DevOps with any framework, and with this type of logic, we can say that using containers is not mandatory for implementing DevOps, but it simplifies the process. For instance, containerization enables teams to work in the same static environment, which makes collaboration more seamless[8]. Furthermore, microservices are containerized separately as previously mentioned, which makes it possible to scale up environments or perform changes related to one specific container, without affecting other services[9].

2. Kubernetes

Kubernetes is a portable and extensible container orchestrator platform, developed by Google and distributed through the Cloud Native Computing Foundation. It provides a variety of functionalities useful when it comes to the automating the management of the lifecycle of containerized software[10,11].

2.1. Architecture

Kubernetes incorporates the use of control planes and nodes, which together make up clusters[12]. The control planes are responsible for distributing tasks over the nodes. Subsequently, the nodes can be seen as worker machines used to run a group of containers sharing some properties, i.e., storage and network. These groups are commonly referred to as pods [12,13].

Moreover, the control planes include a set of components which account for various processes such as accepting requests, scheduling and scaling, among others. These components are the kube-apiserver, etcd, kube-scheduler, kube-controller-manager and cloud-controller-manager. Furthermore, each node includes a set of components that preserve the runtime environment. These components are referred to as the kubelet and kube-proxy, in addition to the container runtime[14]. The architecture is illustrated in *Figure 1*.

[8] Tozzi, Christopher. 2017. What Do Containers Have to Do with DevOps, Anyway? *Container Journal.* January 11. URL: https://containerjournal.com/uncategorized/containers-devops-anyway (Received: 2021-04-26)
[9] Bradley, Tony. n.d. The challenges of scaling microservices. *TechBeacon*. URL: https://techbeacon.com/app-dev-testing/challenges-scaling-microservices (Received: 2021-04-30)
[10] VMware. n.d. What is Kubernetes? URL: https://www.vmware.com/topics/glossary/content/kubernetes (Received: 2021-04-30)
[11] Kubernetes.io. n.d. What is Kubernetes? URL: https://kubernetes.io/docs/concepts/overview/what-is-kubernetes (Received: 2021-04-30)
[12] Red Hat. n.d. What is Kubernetes? URL: https://www.redhat.com/en/topics/containers/what-is-kubernetes (Received: 2021-04-30)
[13] Kubernetes.io. n.d. Nodes. URL: https://kubernetes.io/docs/concepts/architecture/nodes (Received: 2021-04-30)
[14] Ellingwood, Justin. 2018. An Introduction to Kubernetes. *Digital Ocean.* May 2. URL: https://www.digitalocean.com/community/tutorials/an-introduction-to-kubernetes (Received: 2021-04-30)

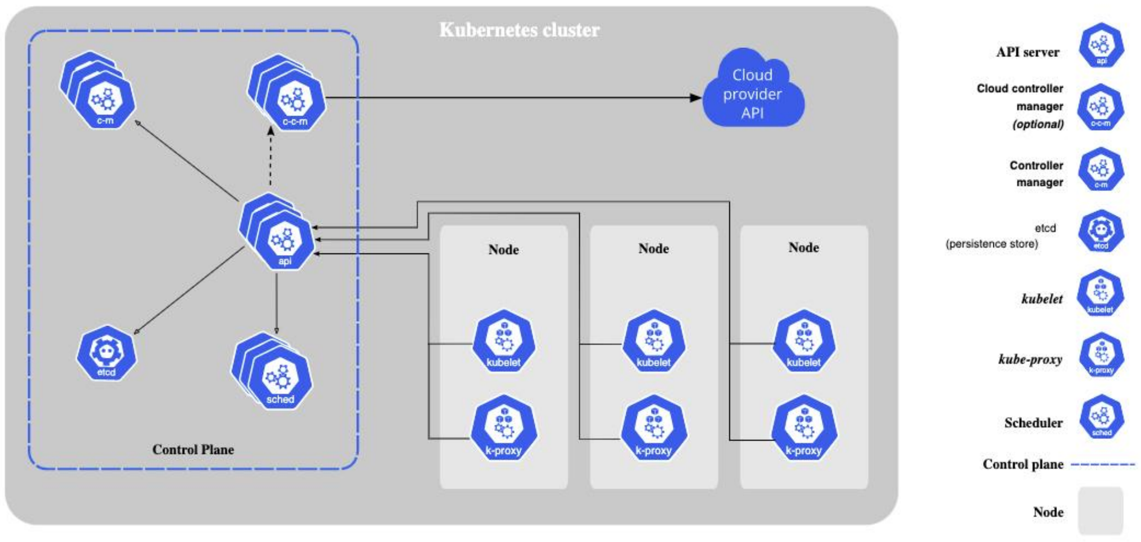

Figure 1. Kubernetes architecture[15]

2.2. Scheduling

The scheduling is a process in which the pods are distributed to the nodes, based on various predefined requirements. The control plane component responsible for executing this process upon the creation of a pod, is the kube-scheduler. An overview of the allocation algorithm is provided below:

1. First, filtering is performed in order to find several nodes that are feasible for allocation of the pod. In case the algorithm returns `NULL`, the scheduling of the pod is postponed until a suitable candidate node exists.
2. Next, the kube-scheduler selects all nodes that passed the filtering. Thereafter, the nodes are rated based on the predefined requirements, in order to determine the optimal pod allocation.

The algorithm results in the pod being allocated to one of the nodes with the maximum rating. The scheduling is followed by another process, in which the result is forwarded to kube-apiserver[16].

3. Nomad

Nomad is a workload orchestrator from HashiCorp that provides cluster management and concurrent shared state scheduling, with support for containerized as well as non-containerized applications[17].

[15] Kubernetes.io. n.d. Kubernetes Components. URL: https://kubernetes.io/docs/concepts/overview/components/ (Received: 2021-04-30)
[16] Kubernetes.io. n.d. Scheduling and Eviction. URL: https://kubernetes.io/docs/concepts/scheduling-eviction/_print (Received: 2021-04-30)
[17] Lärfors, Jacob. 2020. Nomad: The workload orchestrator you may have missed. *Verifa*. September 5. URL: https://verifa.io/insights/nomad-the-workload-orchestrator-you-may-have-missed/ (Received: 2021-04-26)

3.1 Architecture

Nomad incorporates the use of servers and clients in the form of a single binary[18]. The servers constitute the control plane[19], and are responsible for handling jobs, performing evaluations and distributing tasks, i.e., applications in Docker containers[20], to the clients. The clients constitute the platform on which the specified driver for a task, i.e., Docker, Java or Exec, executes[21,22]. The architecture is illustrated in *Figure 2a.*

Each server is allocated to a specific region, in which the clients are stationed in various data centers. This enables the servers to handle huge clusters, where millions of tasks are running on thousands of clients[23]. Furthermore, since multi-region deployment is supported, it is possible to target multiple regions from a single job script[19,24]. This is useful for global companies[19]. Multi-region deployment is visualized in *Figure 2b.*

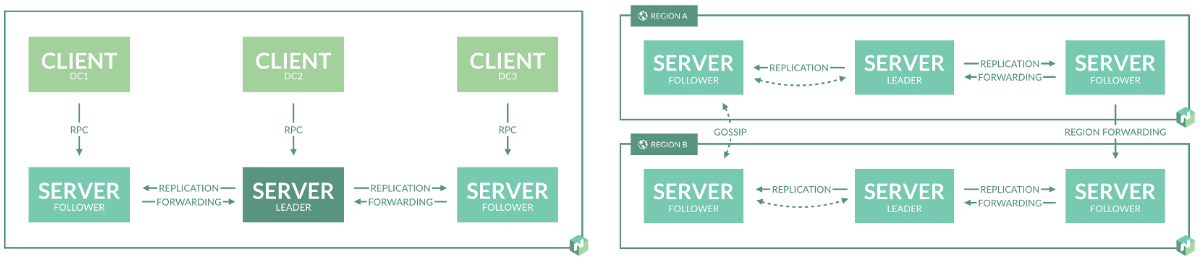

Figure 2a/2b. Nomad architecture/multi-region deployment

3.2 Scheduling

The scheduling is performed using the concept of infrastructure-as-code, following a declarative job script, which specifies the quantity of container instances and provides details regarding the execution of different tasks[25]. Following the registration, deregistration or update of a job, an evaluation is performed, in which the current state is compared in relation to the desired state, that is specified in the job scrip, in order to produce an action plan that specifies to the client how different tasks should be allocated to nodes in the cluster[26,27]. This process is visible in *Figure 3*.

[18] Nomadproject.io. nd. Introduction to Nomad. URL: https://www.nomadproject.io/intro (Received: 2021-04-26)
[19] Gross, Tim. 2020. Multi-Cluster Deployments with HashiCorp Nomad. *HashiCorp.* November 2. URL: https://www.hashicorp.com/resources/multi-cluster-deployments-with-hashicorp-nomad (Received: 2021-04-26)
[20] Fishner, Kevin. 2016. Lessons Learned from Scheduling One Million Containers with HashiCorp Nomad. InfoQ. URL: https://www.infoq.com/articles/scaling-containers-hashicorp-nomad/ (Received: 2021-04-26)
[21] Nomadproject.io. Architecture. nd. URL: https://www.nomadproject.io/docs/internals/architecture (Received: 2021-04-26)
[22] Li, Chang. 2021. A Kubernetes User's Guide to HashiCorp Nomad. *HashiCorp*. February 25. URL: https://www.hashicorp.com/blog/a-kubernetes-user-s-guide-to-hashicorp-nomad (Received: 2021-04-26)
[23] Parsloe, Russ. 2019. HashiCorp Nomad — From Zero to WOW! *Medium*. December 22. URL: https://medium.com/hashicorp-engineering/hashicorp-nomad-from-zero-to-wow-1615345aa539 (Received: 2021-04-26)
[24] HashiCorp Learn. n.d. Multi-Region Deployments. URL: https://learn.hashicorp.com/tutorials/nomad/multiregion-deployments (Received: 2021-04-26)
[25] Ranganathan, Rajagopalan. *A highly-available and scalable microservice architecture for access management.* Aalto University. 2018.
[26] Scheduling in Nomad. URL: https://www.nomadproject.io/docs/internals/scheduling/scheduling
[27] Lozancic, Tomislav. 2017. Microservices with Nomad and Consul. 8 November. *CodeCentric*. URL: https://blog.codecentric.de/en/2017/11/microservices-nomad-consul/ (Received: 2021-04-26)

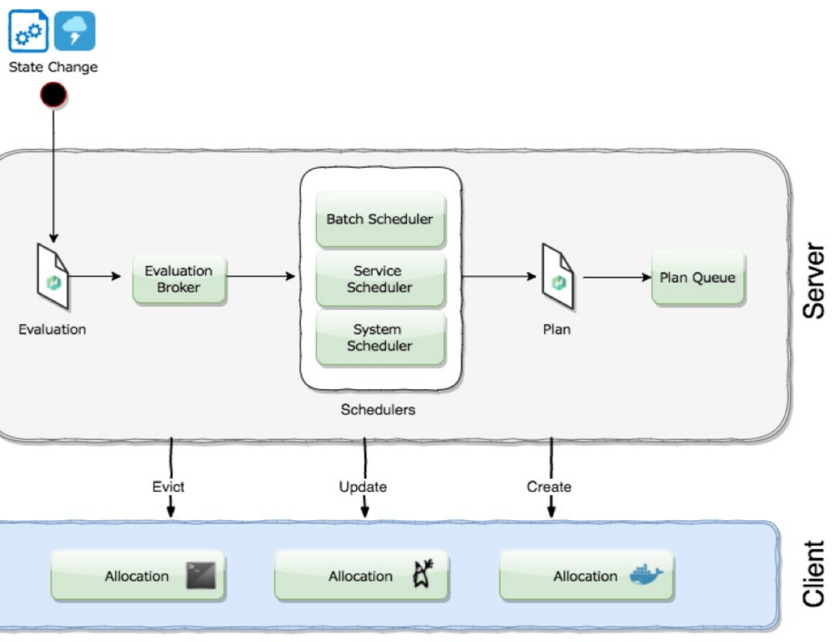

Figure 3. Scheduling in Nomad[27]

Since the concept of concurrency has been implemented, latency for workloads is decreased[28]. Not only can numerous tasks be performed simultaneously, but without the risk of obstruction at large enterprise companies, where multiple development teams submit jobs simultaneously[2].

An overview of the allocation algorithm is provided below:

1. The details of deployment, such as time of deployment for an application and the resources needed are sent as input to the algorithm.
2. The algorithm ensures an optimal placement for each task instance by performing a two-step process[29]:
    - First, all viable nodes are listed, i.e., nodes that fulfill certain constraints
    - Then, various affinity and anti-affinity rules are applied as a measure in order to prevent collocation
3. Furthermore, since the algorithm is bin packing, maximal resource utilization of each individual node is guaranteed. It also simplifies the process of finding available nodes and discontinuing unused clients, all of which contributes to minimizing costs[20].

4. Comparison of Nomad and Kubernetes

This section provides a comparison of the container orchestration tools. According to Matthew Revell, the following aspects are of interest when comparing container orchestrators: architecture, abstraction level and integration with external tools[1]. When comparing the tools with regard to these aspects, the following stands out:

[28] Nomadproject.io. Introduction to Nomad. n.d. URL: https://www.nomadproject.io/intro (Received: 2021-04-26)
[29] Nomadproject.io. Scheduling in Nomad. n.d. URL: https://www.nomadproject.io/docs/internals/scheduling/scheduling (Received: 2021-04-26)

- The installation of Kubernetes is more time and resource consuming, since it includes multiple processes, compared to Nomad which is a single binary[22].
- Both tools are based on a client-server architecture, although the clients and servers are named as control planes and nodes in Kubernetes[22].
- Nomad integrates with multiple drivers in addition to Docker, in contrast to Kubernetes[30].
- Nomad is less complex from an architectural perspective and relies on integration with third-party services for features such as service discovery and secret management[31] (*Figure 4)*. These tools require separate configuration, and their usage raises some difficulties when it comes to maintenance, monitoring and upgrading of the tools[31,32].
- Kubernetes has a powerful API with support for authorization management and a wide range of networking solutions[33,34].
- Both orchestrators offer a built-in rollout and rollback mechanism, and it is possible to auto-revert to a previous version of the job script, if a task fails[35,36] .
- Both orchestrators provide auto-scaling to large-size clusters. Nomad tolerates scheduling of about 3750 containers per second and is scalable for clusters of ten thousand nodes, and Kubernetes appears to be scalable for clusters up to 15 000 nodes[30,37]. However, if you wish to scale a Kubernetes cluster with more than 5000 nodes there are special constraints that have to be fulfilled[38].

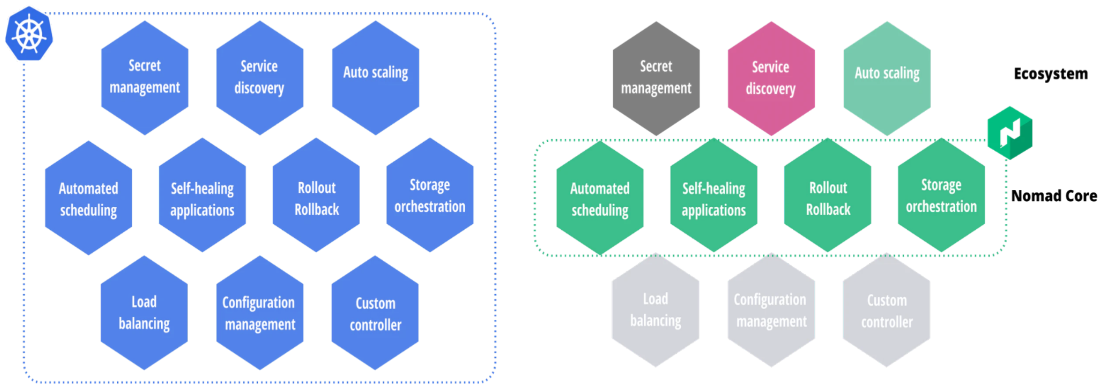

Figure 4. Comparison of services incorporated into the container orchestrators[22]

[30] Nomadproject.io. n.d. Nomad vs. Kubernetes. URL: https://www.nomadproject.io/docs/nomad-vs-kubernetes (Received: 2021-04-30)
[31] Korhonen, Mikko. *Analyzing Resource Usage on Multi Tenant Cloud Cluster for Invoicing*. University of Oulu, 2017.
[32] Kublr Team. 2017. Choosing the Right Containerization and Cluster Management Tool. URL: https://blog.kublr.com/choosing-the-right-containerization-and-cluster-management-tool-fdfcec5700df (Received: 2021-04-30)
[33] Kubernetes.io. n.d. Authorization Overview. URL: https://kubernetes.io/docs/reference/access-authn-authz/authorization/ (Received: 2021-04-30)
[34] Kubernetes.io. n.d. Cluster Networking. URL: https://kubernetes.io/docs/concepts/cluster-administration/networking/ (Received: 2021-04-30)
[35] HashiCorp Learn. n.d. Rolling Updates. URL: https://learn.hashicorp.com/tutorials/nomad/job-rolling-update https://kubernetes.io/docs/concepts/cluster-administration/networking/ (Received: 2021-04-30)
[36] Pallari, Jaakko. 2019. Check your Kubernetes deployments! *Medium*. April 24. URL: https://medium.com/polarsquad/check-your-kubernetes-deployments-46dbfbc47a7c (Received: 2021-04-30)
[37] Long, Rob and Różacki, Maciek. 2020. B*ayer Crop Science seeds the future with 15000-node GKE clusters.* June 23. URL: https://cloud.google.com/blog/products/containers-kubernetes/google-kubernetes-engine-clusters-can-have-up-to-15000-nodes (Received: 2021-04-30)
[38] Guidelines for creating scalable clusters. n.d. URL: https://cloud.google.com/kubernetes-engine/docs/best-practices/scalability#above-5000 (Received: 2021-04-30)

5. Conclusion

In this essay, the concept of container orchestration and its relation to DevOps was introduced. Furthermore, the architecture and scheduling processes used by Kubernetes and Nomad were briefly discussed. The two orchestration tools were compared with regard to their architecture, abstraction level and integration with external tools, in order to provide some guidance in the "orchestration tool jungle". The conclusion was drawn, that although both tools are client-server based, the architecture provided by Nomad is more basic compared to the one provided by Kubernetes. This showcases in the fact that Nomad relies on third party tools for common container orchestration services. Moreover, both tools support large clusters. Hence, Nomad might be a good fit for smaller enterprises that are looking to spend less resources on technical knowledge, although Nomad offers multi-region deployment and can be used for large global enterprises. Whereas Kubernetes is likely a good fit for larger enterprises with access to technical knowledge.