Mastering Containers with OpenNebula Version 1.2 – April 2021 Abstract Application container technologies, like Docker and Kubernetes, are becoming the de facto leading standards for packaging, deploying, and managing applications with increased levels of agility and efficiency. Docker uses OS-level virtualization to deliver software in packages called containers, whereas Kubernetes is a widely used tool for the orchestration of containers on clusters. Although Kubernetes is a powerful tool, it doesn't necessarily work for every single use case nor does it solve all container management-related challenges an organization might face. Kubernetes is a very complex and demanding technology, and other open source alternatives may actually be the best solution for many use cases. OpenNebula offers a simple but powerful approach for running containerized applications and workflows by directly using the Docker official images available from the Docker Hub and running them on lightweight Firecracker microVMs that provide an extra level of efficiency and security. This solution combines all the benefits of containers with the security, orchestration, and multi-tenant features of a solid Cloud Management Platform but without adding extra layers of management, thus reducing the complexity and costs, compared with Kubernetes or OpenShift. You can also run your containers on a cloud environment based on LXC system containers if you need full bare-metal performance and isolation is not a requirement. For those cases where Kubernetes is required or is the best fit, OpenNebula brings support for the deployment of Kubernetes clusters through a CNCF-certified Virtual Appliance available from the OpenNebula Public Marketplace or through the K3s lightweight distribution for resource-constrained and edge locations. Last but not least, OpenNebula also offers integration with other popular orchestration engines such as Docker Machine, Docker Swarm, and Rancher. Contents 1. A High Level View of Containers 2. What is OpenNebula? 3. Container Workflows with OpenNebula 4. Using Kubernetes with OpenNebula 5. Other Approaches 6. Ready for a Test Drive? 7. Conclusions Glossary ACL Access Control List CMP Cloud Management Platform K8S Kubernetes VDC Virtual Data Center VM Virtual Machine VMM Virtual Machine Monitor Mastering Containers with OpenNebula 1. A High Level View of Containers Figure 1. Available solutions for container orchestration with OpenNebula. The Container Revolution Information and Communication Technologies have evolved at a really fast pace over the past few years, dramatically changing the way information systems and applications are built. Software development has brought along many changes and revolutions, now allowing people to focus mainly on the business applications. The key drivers of this shift have been (1) the ability to package and run applications anywhere regardless of the underlying computing architecture, and (2) the means of keeping applications isolated from each other to avoid security risks and interference during operations or maintenance processes. Keeping applications isolated on the same host or cluster can be difficult due to the packages, libraries, and other software components that are normally required to run them. Hardware virtualization was a solution to this problem, since applications could be kept apart on the same hardware

by using Virtual Machines. Packaging an application within a VM also allowed it to run on any infrastructure that supported virtualization, which provided a lot of flexibility to the whole concept. However, Virtual Machines come with some serious limitations: moving them around is not that easy, since they are typically quite heavy and there are always difficulties associated with maintaining and upgrading applications running within a VM. In recent years, we have all witnessed how container technologies have revolutionized the way enterprise and distributed applications are being developed and deployed. Containers clearly offer a more portable and flexible way of packaging, maintaining and running applications. They allow admins to deploy, move and replicate workloads more quickly and easily than using Virtual Machines. While containers as a concept have been around for a while, Docker was the technology that introduced several crucial changes to the existing container technology, making containers more portable and flexible to use. This resulted in a turning point towards the adoption of containerization and microservices in software development (e.g. cloud-native development). Docker gave us an easy way to create container-based applications and to package them in portable images containing the specifications for the software components the container will run. Docker's technology brought cloud-like flexibility to any infrastructure capable of running containers, with their container image tools allowing developers to build libraries of images, compose applications from multiple images, and launch those containers and applications on local and remote infrastructures alike. Version 1.2 April 2021 Page 2 of 14 Mastering Containers with OpenNebula Orchestrating Containers Nowadays, many companies have embraced a cloud-native paradigm in developing applications and have shifted from a "monolithic" approach to a microservice approach. While deploying a single container can be an easy task, things get a bit more complicated when deploying multi-container applications on distributed hosts given that in these cases a Docker Engine alone is not enough. This is where container orchestrators (like Kubernetes or Docker Swarm) play an important role in scheduling containers to run on different servers, moving containers to a new host when the host becomes unhealthy, restarting containers when they fail, managing overlay networks to allow containers on different hosts to communicate, orchestrating storage to provide persistent volumes to stateful applications, and so on. However, container technologies (e.g. Docker, Kubernetes) also come with some serious limitations, such as for example security (application containers share the kernel OS) and multi-tenant environments. In order to provide a multi-tenant and secure environment to deploy containerized applications, one has to provision different "virtual environments" to each user or group of users, typically by deploying several isolated Kubernetes clusters on top of a Cloud Management Platform. The CMP is then responsible for managing and orchestrating the underlying virtual resources (i.e. Virtual Machines, virtual networks and storage) that are used by the different Kubernetes deployments that are in charge of scheduling application containers within those isolated environments. This approach adds an extra control

layer that ends up increasing management complexity, resource consumption, and operational costs. Running Containers on MicroVMs But what if we could remove one of those layers? Cloud Management Platforms such as OpenNebula have been implemented with multi-tenancy and security by design, providing already powerful orchestration features but for VM-based applications (e.g. networking, storage blocks, high availability with live/cold migrations, etc.). Now Firecracker has been incorporated into OpenNebula as a new supported virtualization technology. This microVM technology, developed by Amazon Web Services (AWS) and widely used as part of its Fargate and Lambda services, has been especially designed for creating and managing secure, multi-tenant container and function-based services. By taking this step, OpenNebula has managed to bridge the gap between two technological worlds, leaving behind the old dilemma of whether to use containers—lighter, but with weaker security—or Virtual Machines—with strong security but high overhead. By adopting the increasingly popular approach of running microservices based on containerized applications, and thanks to its seamless integration with Docker Hub, OpenNebula has now become a powerful alternative to deploy and orchestrate containers as secure and fast Firecracker microVMs. Section 2 of this white paper offers a general introduction to OpenNebula, while Section 3 provides insight into the technical foundations of OpenNebula's new native model for container orchestration. No single size fits all and there are use cases that still require the running of a Kubernetes cluster, including those cases in which your target application has been defined as a Helm Chart and so you may need to use Kubernetes on your OpenNebula cloud. For those situations, as described in Section 4, OpenNebula provides support for the deployment of Kubernetes clusters through a Virtual Appliance available from the OpenNebula Public Marketplace. Other cases may require alternative orchestration engines like Docker Swarm or Rancher. The integration with these tools is described in Section 5 of this white paper.