EPiC Series in Computing

Volume 78, 2021, Pages 69–76

Proceedings of the European University Information Systems Conference 2021

JupyterHub on an on-premises cloud – a special focus on GPU Accelerated Machine Learning and 3D Visualization

Markus Blank-Burian[1], Jürgen Hölters[1], and Raimund Vogl[1]

WWU Münster, Germany
{blankburian,holters,rvogl}@uni-muenster.de

Abstract

At the IT department of the University of Münster (WWU IT) we build a private IaaS cloud based on OpenStack and Kubernetes (WWU Cloud). This cloud provides a generic platform for data storage and service hosting. WWU IT operates a JupyterHub on WWU Cloud for use in research and education. Researchers have access to virtual GPUs from their Jupyter sessions. These may be used to compile and natively run CUDA accelerated code, e.g. for machine learning. Using VirtualGL, we also provide an accelerated X server in Jupyter sessions. X11 applications are then accessible from the browser using noVNC.

1 Introduction

Jupyter notebooks [5, 8] are a defacto standard for scientific research. They provide an easy to use interface for interactive, reproducible data analysis and visualization. Lately, the traditional notebook interface has been replaced by JupyterLab [4]. This new user interfaces is easy to navigate by using well-known elements from desktop applications and has an extendable plugin architecture.

At the IT department of WWU Münster (WWU IT), we operate a JupyterHub using KubeSpawner on a Kubernetes cluster [3]. This Kubernetes cluster itself runs on WWU Cloud, a private IaaS cloud based on OpenStack [9] and part of the RDI-NRW project [10]. The JupyterHub can be used by students, in teaching and by researchers. We provide a selection of notebook images containing standard applications directly available on the JupyterLab launcher page.

As machine learning became a major topic in research and teaching, it was requested make GPU resources available in our Jupyter sessions. In addition, medical researchers needed a solution for accelerated remote data visualization using GPUs. In response to these requests, we setup the necessary infrastructure providing virtual GPUs within Jupyter sessions.

In the following sections, we will describe our complete cloud stack in a bottom-up approach with focus on GPU acceleration, starting with a quick overview on Kubernetes, followed by WWU Cloud, WWU Kube and finally JupyterHub (Figure 1).

S. Bolis, J.-F. Desnos, L. Merakos and R. Vogl (eds.), EUNIS 2021 (EPiC Series in Computing, vol. 78), pp. 69–76

2 Kubernetes

We use Kubernetes as a platform to deploy services on bare metal and in virtual machines on cloud resources. Kubernetes is a container orchestration engine [3], which is configured declaratively using a REST API. Originally created by Google, Kubernetes has become a widely used platform for operating services on the internet. The objects, created using the Kubernetes REST API, represent a desired state, which is periodically reconciled with the actual state. YAML [2] is the preferred markup language to write object representations in API calls, as it is more easy to write and can be converted directly to JSON. One important feature of Kubernetes as a container orchestration engine is its integrated high availability and resiliency towards hardware failures. Services are automatically restarted on hardware or software failures and user requests are load-balanced between all healthy processes. Monitoring and alerting solutions from the cloud native ecosystem integrate nicely with Kubernetes clusters, providing insights into resource usage and quick notifications on errors.

Applications for use in Kubernetes have to be packaged into container images, typically containing an operating system image without kernel as well as the application and all its dependencies. Many modern images contain only a single statically linked binary, thereby reducing the potential attack surface. Multiple containerized applications may run on the same host with only minimal interaction, as they each use an independent filesystem. The Kubernetes "kubelet" daemon, running on a single node, starts an application with the help of a container runtime, which is responsible for downloading and unpacking the container image, setting up the container and starting the application. This architecture makes it possible to quickly update a service, by uploading a new application image or rolling back to a previous version.

Due to these many great features, using Kubernetes as base platform for deploying our cloud platform and services greatly improves stability, manageability and observability.

3 WWU Cloud

The WWU Cloud is a private IaaS (Infrastructure as a Service) cloud platform, based on OpenStack and Ceph, for use by WWU IT and researchers. Ceph is a storage system, designed for exabyte capacity, and is comprised of an object store (RADOS), a block device interface (rbd), a filesystem (CephFS) and an S3 gateway (RadosGW). The WWU Cloud exposes the last two directly to users and block devices in form of virtual volumes, which can be attached to virtual machines. OpenStack provides an interface (API and Web) for self-service creation of virtual infrastructure (virtual machines, virtual volumes, virtual networks and network shares).

In WWU Cloud, Projects with fixed quotas on virtual hardware can be requested free of charge. The actual hardware is overcommitted (15x for CPUs and 1.5x for RAM), making the cloud most ideal for services or interactive usage. Once a project in OpenStack is created, virtualized infrastructure can be created on demand by the user. The hardware resources of the WWU cloud consist of a heterogeneous mix of hyperconverged nodes, providing compute and storage resources as well as GPUs for use in data visualization and machine learning. The hyperconverged architecture allows for easy expansion and maximum resource utilization.

To orchestrate OpenStack, Ceph and other cloud platform services, we use a baremetal Kubernetes cluster. The deployment method of this cluster is currently being replaced with Cluster API [1], which provides a convenient way to provision baremetal nodes using operating system images. The Metal3.io backend for Cluster API works together with OpenStack Ironic to perform the actual baremetal provisioning. We deployed Cluster API into a Kubernetes deployment cluster, from where we manage multiple deployment regions. New operating system

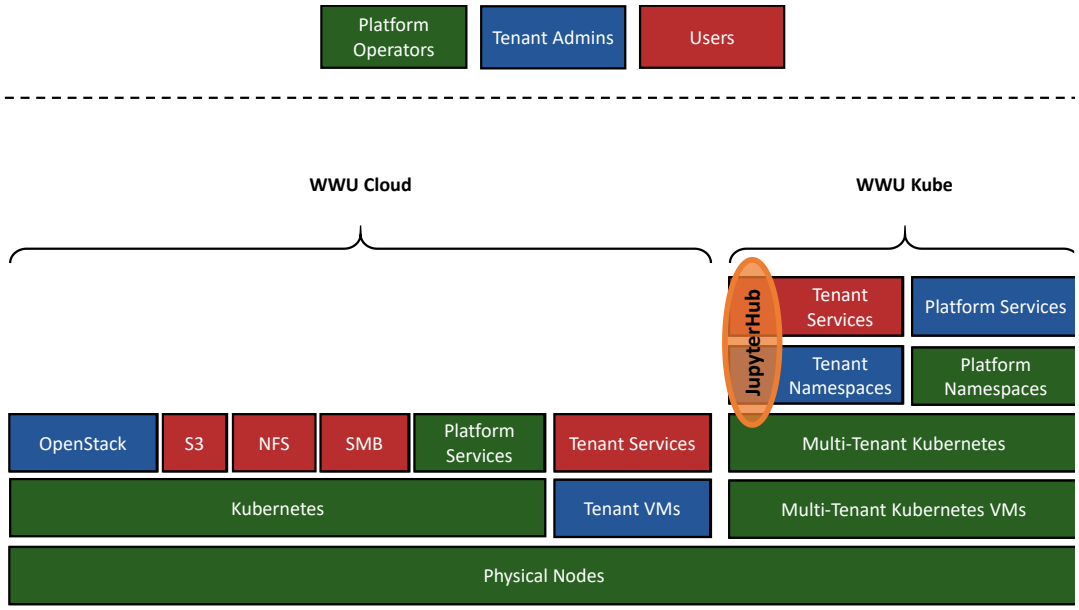


Figure 1: Graphical representation of the software architecture for WWU Cloud and WWU Kube. As a consequence of our hyperconverged design, all cloud platform components including storage and virtual machines run on the same hardware. JupyterHub runs as a service in WWU Kube, which is a Kubernetes cluster running in virtual machines on WWU Cloud.

images are built and uploaded in a fully automated CI/CD workflow. The resulting images can then be deployed using Cluster API on any new or existing node using a single command. During an update, the nodes are automatically drained by Cluster API, stopping all running containers, but not the virtual machines. We wrote a small controller to migrate all virtual machines to other hosts, once a node is drained, thereby fully automating node updates without service interruption.

The GPUs are virtualized as VFIO mediated devices "mdev" using NVidia GRID technology. In our current setup, device type selection in Nova is limited to a single device type per card. In the future, this limitation can be avoided with OpenStack Cyborg, making it possible to dynamically select an arbitrary vGPU type for each VM.

Without Cyborg, OpenStack makes custom GPU resources available to virtual machines as a single untyped "VGPU" resource, making it impossible to distinguish between different GPU types. As we only use a single GPU type per node, we have worked around this limitation by specifying the GPU type as node trait, which can be added administratively on each node. This makes it possible to define multiple vGPU flavors, individually selectable by newly created virtual machines.

4 WWU Kube

Within virtual machines in the WWU Cloud, WWU IT runs a multi tenant Kubernetes cluster called WWU Kube. Users of WWU Kube may use this platform to operate arbitrary services. At WWU IT for example, we operate JupyterHub and our scalable Jitsi deployment on WWU

Kube.

Kubernetes in WWU Kube nodes are organized in multiple groups: control plane, nodes, worker nodes and worker nodes with vGPU. Control plane and node VMs have hardware node anti-affinity. While the former group is intended to run only platform critical services, the latter group is suited for all types of services requiring high availability. Worker nodes without node anti-affinity are used for Jupyter sessions, as these run in single pods without failover capabilities. This Kubernetes cluster is provisioned using Cluster API, allowing the cluster to automatically scale up more nodes based on the actual load for each node group. Specifically our JupyterHub deployment benefits from autoscaling, as compute resources and especially GPU resources can be allocated on demand.

Nodes with vGPUs are automatically labeled using the "node-feature-discovery" and "nvidia-device-plugin" tools. These labels can be used by Kubernetes pods to select the nodes for an application and thereby a gpu type. The amount of GPUs for a pod is specified via Kubernetes resource requests and limits. The device inodes and environment variables are automatically injected into the container, so the application can transparently access the virtual device.

As users with Jupyter sessions gain terminal access within containers in WWU Kube, special emphasis has to be put on security. We employ cilium network policies to restrict access within the cluster and Kubernetes pod security policies as well as OpenPolicyAgent Gatekeeper policies to restrict permissions on newly spawned pods and enforce node selectors. For enhanced observability we run Falco, an eBPF based security scanner to detect privilege escalations and exploits. To prevent resource exhaustion, we impose detailed quotas on the JupyterHub project namespace.

Services within WWU Kube are deployed using GitOps with ArgoCD. GitOps is a deployment method, where the desired state of an application is represented in a git repository [6]. ArgoCD is implemented as a controller, which periodically reconciles the actual state with the desired state. An application may thus be updated or rolled back using a single git commit. Multiple application states (e.g. production and development) are realized using branching. We use an apps-of-apps pattern to recursively deploy all system applications as well as WWU IT deployed applications in the cluster. For testing purposes, we have additional development and staging clusters, which are also managed using branching in git.

5 JupyterHub

JupyterHub is a webservice, providing jupyter notebook sessions hosted on dedicated server hardware. WWU IT operates a JupyterHub deployment on WWU Kube, available for all students and employees of WWU, primarily used by researchers and for courses. JupyterHub is configured to use "KubeSpawner" to spawn new instances as Kubernetes pods. After a successful login, users are presented with a dialog for selecting the cpu and ram limits as well as the notebook image and vGPU support. WWU IT provides a small selection of notebook images, containing commonly used software for data analysis and visualization.

JupyterLab is used as the main user interface, providing a launcher interface to start applications, notebooks and terminal sessions. Combined with a treeview for file selection, git integration and a tabbed interface for opening multiple notebooks, JupyterLab provides a complete virtualized working environment in the browser. Our file storage in WWU Cloud as well as our HPC cluster filesystems are directly available from JupyterLab sessions, making it ideal to quickly analyze and visualize results from computations.

Apart from notebooks, we offer browser based applications like "code-server" (a VSCode

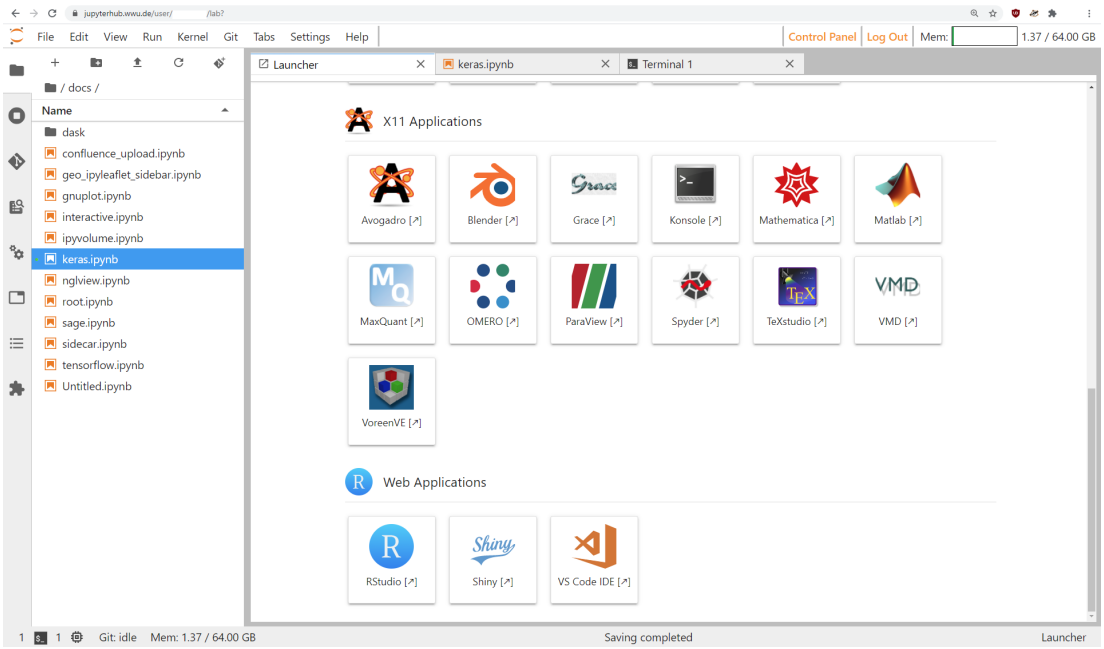


Figure 2: JupyterLab Launcher. In addition to traditional notebooks, X11 and Web applications or a command line terminal can be started from the user interface.

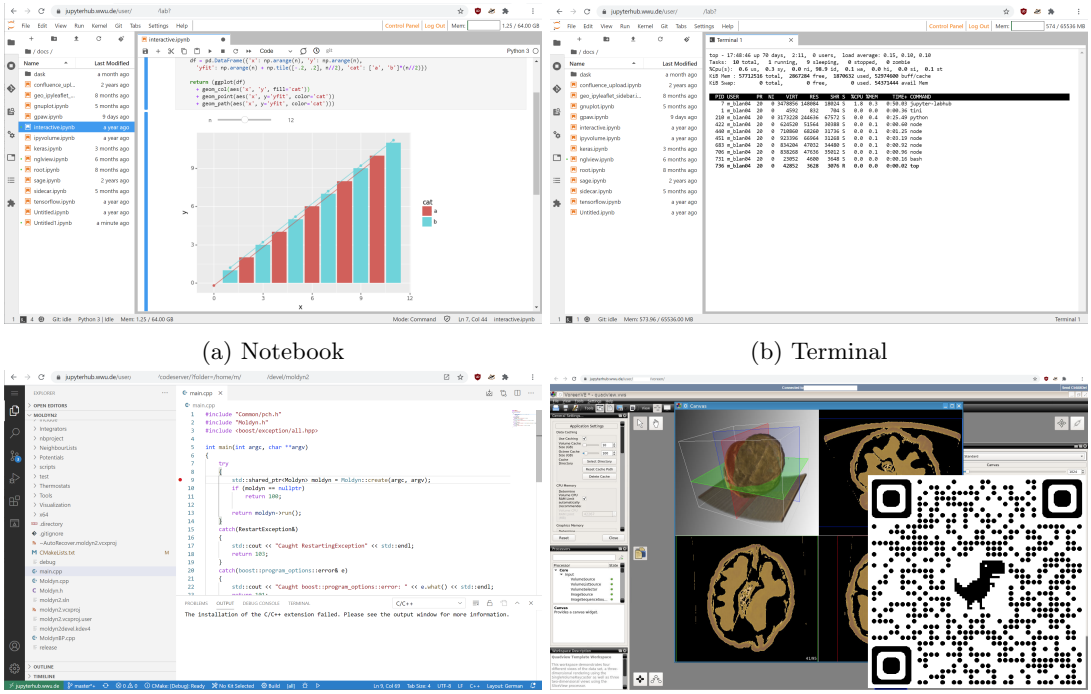
fork) and R Studio. This is realized using websockets and port-forwarding in JupyterLab through the "jupyter-server-proxy" extension. Both web applications run natively in the browser and are therefore very responsive.

We also offer X11 applications in JupyterHub sessions, realized by using noVNC, also using websockets and port-forwarding. noVNC is a browser based remote visualization software based on the VNC protocol. The X11 applications are rendered on an Xorg server in the server side Jupyter sessions. The output is compressed and transferred to the browser, where it is displayed. In case, the session is started with vGPU support, we configure an Xorg server with VirtualGL in a sidecar container, so that multiple applications in a session share a single GPU accelerated framebuffer. X11 Applications see the full set of GLX extensions provided by the native driver and use server based accelerated rendering. At WWU IT, we can therefore offer high end GPU acceleration for data visualization, with no additional requirements on the client side.

When vGPU support is selected, the virtual GPUs are also directly accessible from within the Jupyter sessions. CUDA applications can be compiled and executed natively, e.g. for machine learning tasks. We also integrated major machine learning frameworks (TensorFlow, Kiras, PyTorch) into our notebook images.

6 Discussion

Using open source building blocks, we have implemented a complete private IaaS cloud platform. The Kubernetes container orchestration engine has proven to be a valuable tool in service deployment, from bare metal to clusters of virtual machines. Through the integrated high availability features of Kubernetes, metric based alerting tools and high levels of automation, we



(a) Notebook (b) Terminal

(c) VS Code (d) Voreen

Figure 3: (a) The standard notebook interface of JupyterLab, (b) the standard terminal of JupyterLab, (c) VS Code fork "code-server" as web application and (d) Voreen [7], an X11 3D visualization application, started inside a JupyterHub session. A video demonstration showing server-side rendering can be found at https://radosgw.public.os.wwu.de/wwuit-sys/videos/voreen.mp4.

built a very robust system, which is easy to maintain and requires hardly any human intervention during normal operations. Updates and bugfixes can be tested automatically through the use of CI workflows and be integrated swiftly into staging or production systems using GitOps workflows.

The WWU Cloud as well as the WWU Kube are already used in plenty of projects, most of which are administered by researchers. Our JupyterHub service is primarily used by students, as a basic working environment to learn programming or data analysis. The GPU support in JupyterHub is primarily used by researchers, requiring either remote data visualization or machine learning capabilities.

7 Acknowledgements

- We thank MKW NRW for funding the RDI-NRW project.

References

[1] Cluster api. https://github.com/kubernetes-sigs/cluster-api.

[2] Oren Ben-Kiki, Clark Evans, and Brian Ingerson. Yaml ain't markup language (yaml™) version 1.1. *Working Draft 2008-05*, 11, 2009.

[3] Brendan Burns, Joe Beda, and Kelsey Hightower. *Kubernetes: up and running: dive into the future of infrastructure*. O'Reilly Media, 2019.

[4] B Granger and J Grout. Jupyterlab: Building blocks for interactive computing. *Slides of presentation made at SciPy*, 2016.

[5] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian E Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica B Hamrick, Jason Grout, Sylvain Corlay, et al. *Jupyter Notebooks-a publishing format for reproducible computational workflows.*, volume 2016. 2016.

[6] Thomas A Limoncelli. Gitops: a path to more self-service it. *Communications of the ACM*, 61(9):38–42, 2018.

[7] Jennis Meyer-Spradow, Timo Ropinski, Jörg Mensmann, and Klaus Hinrichs. Voreen: A rapid-prototyping environment for ray-casting-based volume visualizations. *IEEE Computer Graphics and Applications*, 29(6):6–13, 2009.

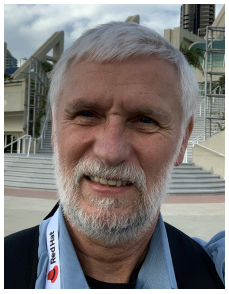
[8] Jeffrey M Perkel. Why jupyter is data scientists' computational notebook of choice. *Nature*, 563(7732):145–147, 2018.

[9] Tiago Rosado and Jorge Bernardino. An overview of openstack architecture. In *Proceedings of the 18th International Database Engineering & Applications Symposium*, pages 366–367, 2014.

[10] Raimund Vogl, Dominik Rudolph, and Anne Thoring. Bringing structure to research data management through a pervasive, scalable and sustainable research data infrastructure. In *The Art of Structuring*, pages 501–512. Springer, 2019.

8 Biographies

M. Blank-Burian is a research assistant at the it department of the University of Münster (WWU IT) and lead cloud architect of the WWU Cloud. He graduated from University of Münster (Germany) in 2013 with degrees in both physics and computer sciences. In 2018, he received his Ph.D. in theoretical physics. His research focuses on private clouds and cloud native technology. More info: https://www.uni-muenster.de/forschungaz/person/17330



J. Hölters is the head of the department for systems at it department of the University of Münster (WWU IT). More info: https://www.uni-muenster.de/forschungaz/person/8172

R. Vogl holds a Ph.D. in elementary particle physics from the University of Innsbruck (Austria). After completing his Ph.D. studies in 1995, he joined Innsbruck University Hospital as IT manager for medical image data solutions and moved on to be deputy head of IT. He served as a lecturer in medical informatics at UMIT (Hall, Austria) and as managing director for a medical image data management software company (icoserve, Innsbruck) and for a center of excellence in medical informatics (HITT, Innsbruck). Since 2007 he has been director of the it department of the University of Münster (WWU IT, Germany). His research interests focus on management of complex information systems and information infrastructures.