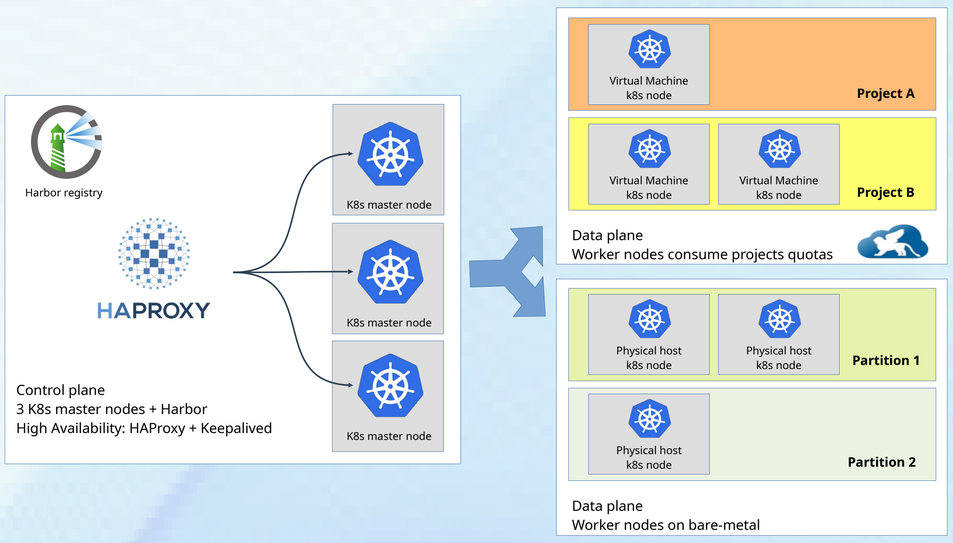
The CloudVeneto's Container-as-a-Service ecosystem

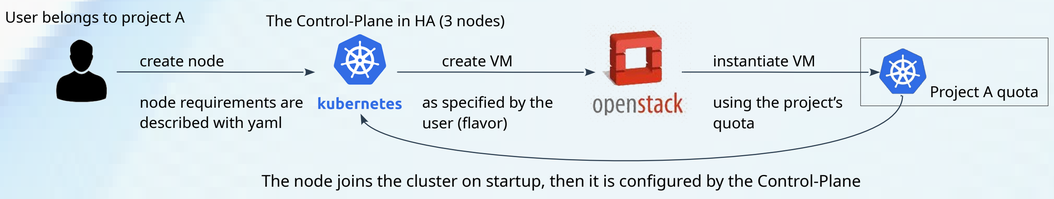
F. Fanzago, M. Sgaravatto, S. Traldi, A. Troja, M. Verlato, L. Zangrando – INFN-PD
D. Lupu, D. Marcato – INFN-LNL

As part of the innovative CloudVeneto framework, we have developed an advanced Container-as-a-Service (CaaS) tailored for different research areas. Our aim is to provide a secure, centrally managed Kubernetes service, alleviating users from administrative burdens and optimizing the use of Cloud resources. Our solution effectively meets the requirements of user communities like Quantum, Isolpharm, and SPES, showcasing its adaptability within the CloudVeneto ecosystem. Moreover, we've demonstrated the ability to offload workload from remote Kubernetes clusters to our CaaS service using the interLink implementation, which extends the Virtual Kubelet concept.

Our Container-as-a-Service solution

Unlike Kubernetes-as-a-Service (KaaS), where users have to manage their own clusters, the Container-as-a-Service (CaaS) model offers a fully centrally managed orchestration platform in the cloud. Users can simply run their Linux containers to the cloud provider, and they don't have to deal with administrative tasks. Our CaaS solution, built on Kubernetes, combines simplicity with user control over private cloud resources. We provide a centralized Kubernetes Control Plane without worker nodes by design, allowing users to create and customize nodes using resources from CloudVeneto within their OpenStack projects. These nodes, deployed as VMs, are fully integrated into the cluster. Users have the flexibility to keep nodes private or share them within the project (namespace), maintaining control while delegating deployment and monitoring tasks to CaaS, thereby easing administrative burdens.





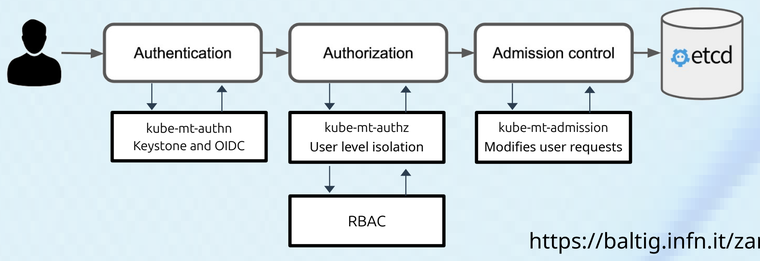
Moreover our architecture enables administrators to deploy nodes on bare-metal and dynamically scale up to handle workload spikes using cloud resources.

kube-multitenancy

While Kubernetes inherently supports multi-tenancy, there are several challenges in achieving strict isolation at both the user and node levels. Users within the same namespace may impact each other, and running pods on shared nodes can pose security and performance risks. To mitigate these issues, we enhanced the Kubernetes security layer by introducing resource ownership concepts and implementing automated pod placement on dedicated nodes based on predefined policies. This ensures that users can only operate on their own resources and have the flexibility to choose whether to deploy their pods on private or shared nodes within their namespace.

The kube-multitenancy architecture enhances Kubernetes for multi-tenancy management. It comprises three key webhooks:

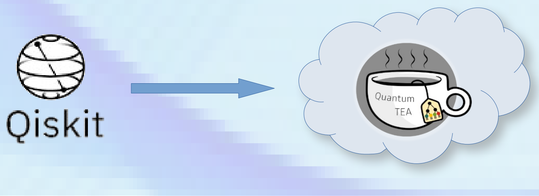
- **kube-mt-authn**: validates user identities for secure resource access (OIDC or Keystone)
- **kube-mt-authz**: enforces access policies to regulate user actions within the namespace
- **kube-mt-admission**: enables admission control, customizing resource requests based on predefined rules.



https://baltig.infn.it/zangrand/kube-multitenancy

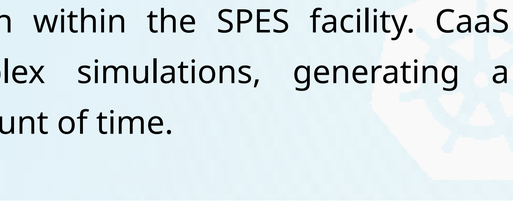
Quantum Computing

The "Quantum Computing and Simulation Center" (QCSC) project utilizes our CaaS service to execute quantum circuits within CloudVeneto. QCSC has developed the QuantumTEA cloud platform, seamlessly integrated into the CloudVeneto ecosystem. This platform empowers Unipd students to delve into quantum computing, enabling them to define and run their own circuits effortlessly using Qiskit, without requiring specific knowledge of the underlying architecture.



Radiopharmaceutical research

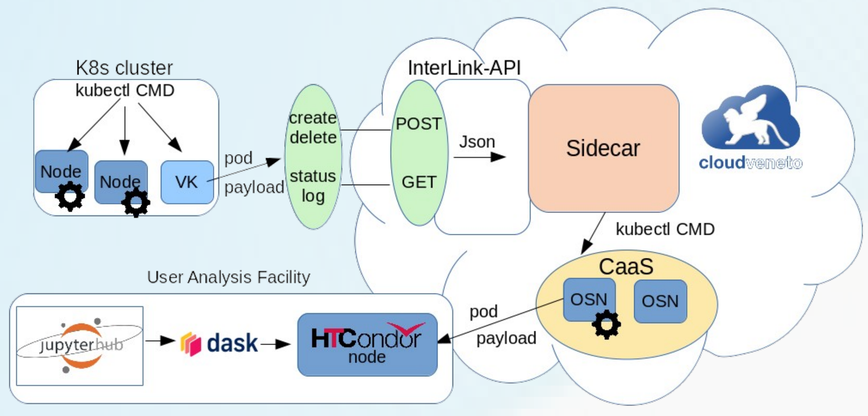
CaaS has significantly improved the accessibility of CloudVeneto resources for projects such as SPES and ISOLPHARM. It has simplified tasks such as deploying and configuring Kubernetes clusters, allowing researchers to efficiently run Monte Carlo simulations for radiological exposure assessments or assessing Ag111 production within the SPES facility. CaaS enabled the parallel execution of complex simulations, generating a considerable number of events in a short amount of time.



Offloading

Offloading allows the execution of containerized code across distributed and heterogeneous computational environments, leveraging specialized hardware from remote computing centers. Referring to the interLink implementation in the interTwin project, the key components are:

- **cluster Kubernetes with a Virtual Kubelet (VK):** VK acts as a 'virtual node' which distributes of user workflows on external resources.
- **interLink-API:** it serves as a bridge between the VK and remote resources.
- **interLink-sidecar:** it receives REST API calls from interLink-API, translating them into commands specific to local resources for user payload execution.



From an analysis facility users can create a dask cluster within HTCondor nodes deployed through offloading.

To interact with our CaaS, we developed a custom interLink sidecar. It incorporates a Flask server and translates the calls into K8s commands, to be executed and managed on CaaS nodes. We have validated this solution through two primary use cases: the CMS High Rate Analysis Facility and the AI-INFN platform.