DEL063 - Continuous Integration of Models and Data

Cloud Notary

In our field of hydrodynamic modelling, it is common to have sensitive and/or large and expensive datasets, which are vital to the business and treated as secret. These datasets however should still be accessible to downstream algorithms and (external) models to analyze and improve these models. In other words, Party A has data x and Party B has model y and both do not want to or cannot share their property. However, they are both interested in the result z = operation(x,y). We envision the use of cloud computing as a facilitator for the operation.

An example of such a cooperation is Van Oord and Deltares. Van Oord has Buoy data which could be useful to improve the Global Tide and Surge Model (GTSM) from Deltares. Both parties cannot share their property but are interested in the improved model results. This use case is taken to create an archite cture in Microsoft Azure as described below.

Key concepts

In our brainstorming sessions, a few key concepts came up. First, that both parties still need to trust both the data and the algorithm to be valid and to do only what they need to do and nothing else. Therefore, a legal contract should be drawn up. Second, the environment that will hold both the data and the algorithm should be transparent as possible, i.e., it should be clear what it will do beforehand. This is best solved by presenting the infrastructure as code beforehand. Third and last, the envisioned operation should run as a functional account that ideally prevents any access by both parties.

- Legal contract in which both parties trusts both data and algorithm.
- Infrastructure as code, transparent how the data and algorithm will run.
- Use functional accounts to prevent user tampering.

Continuous integration and continuous deployment (CI/CD), normally used to test and deploy code on each commit, is a suitable candidate to provision a secure cloud environment. Using Github and Azure DevOps (both from Microsoft) we can store the environment as code and setup a cloud environment. In the Azure DevOps setup, a service level user is created that will gain access to both the Deltares Azure environment as the van Oord Azure environment, but without either parties seeing those keys or gaining access themselves.

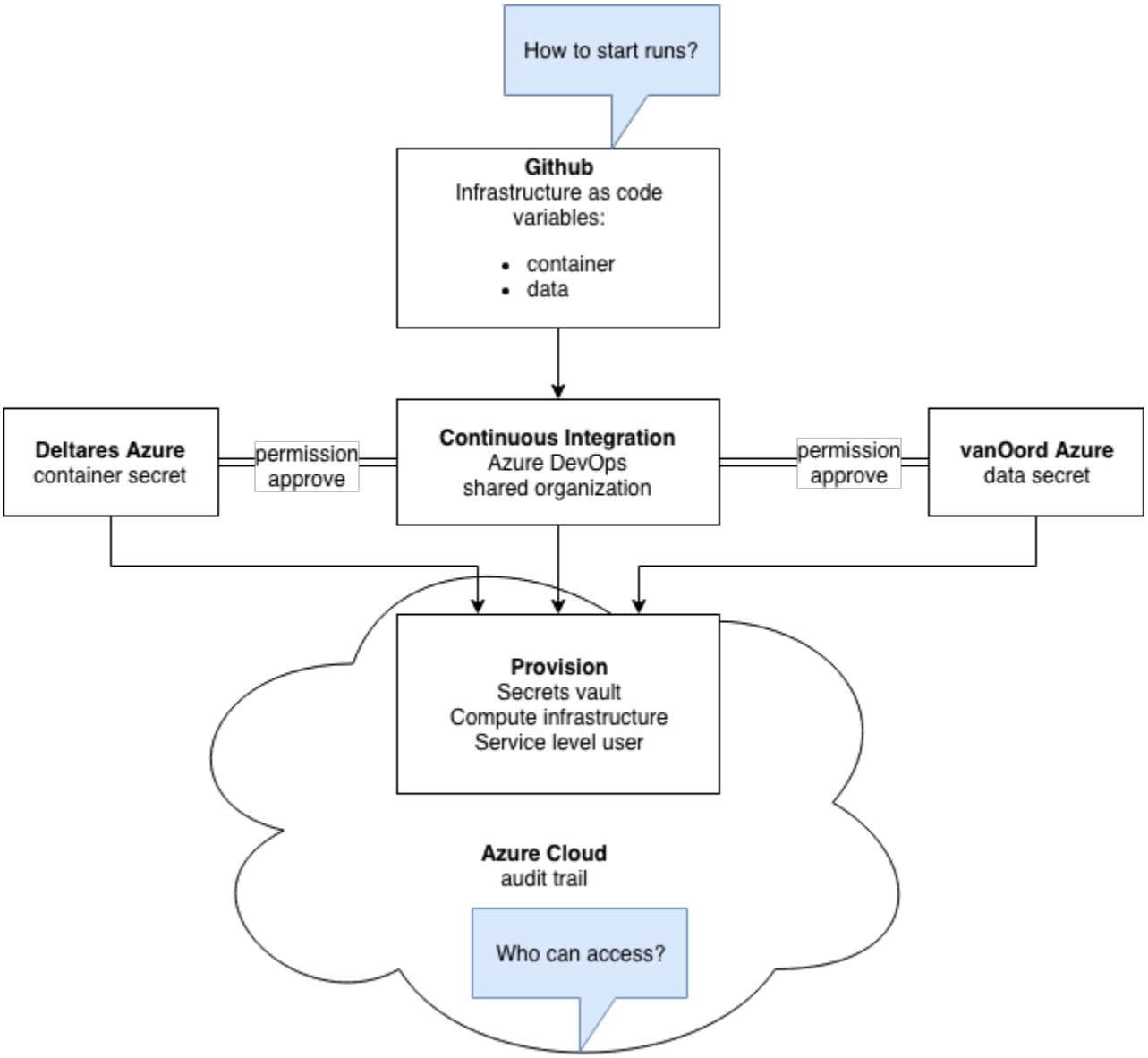


Figure 1 Cloud notary architecture in Azure

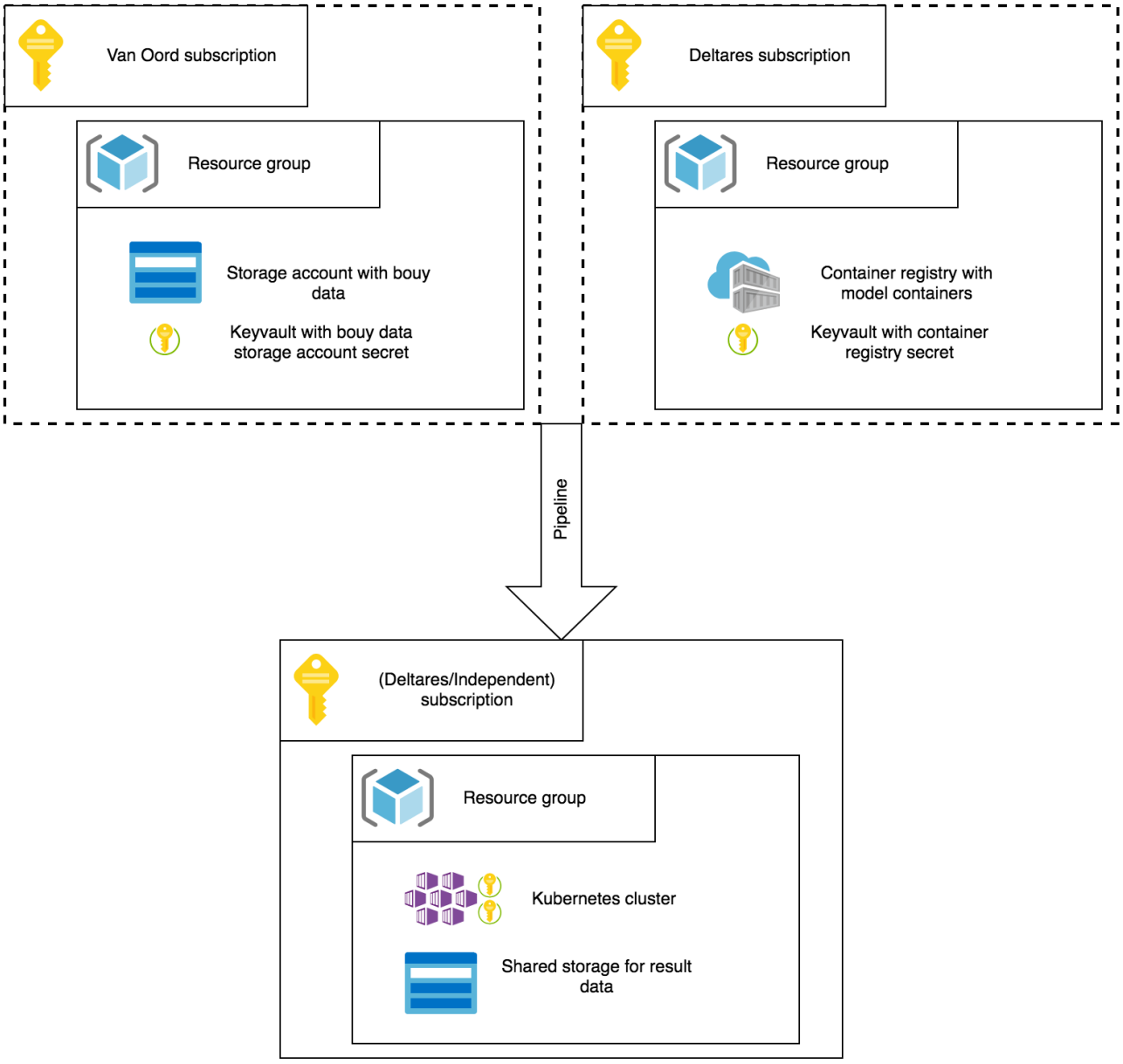
Architecture

The implementation expects an initial setup at Microsoft Azure for the involved parties. This is illustrated in Figure 1.

Van Oord has a resource group within their subscription with a storage account and a key vault. The storage account contains their secret dataset, and the key vault stores a key which provides access to the storage account.

Deltares has a resource group within their subscription with a container registry and a key vault. The container registry contains their container with numerical model and in the key vault a key is stored which provides access to the storage account.

This initial setup is the input for an Azure DevOps pipeline. This pipeline creates a Kubernetes cluster with the keys stored in secrets. This Kubernetes cluster is then used to start a workflow which brings data and model together. A more detailed description of the pipeline is given next.



The actions in the pipeline are illustrated in the image below. A service account from both Van Oord and Azure are used in the pipeline for access to the relevant items in the recourse groups as described above.

- Several keys are extracted from the Deltares key vault

- Access key to a storage account with input data for the model.
- Access key to the container registry in Microsoft Azure for the pre- and post-processing containers.
- Access key to Docker hub delft3dfm registry for the model container

- Service principal for the Kubernetes cluster is extracted from the Deltares key vault. This service account will be used to setup the cluster.
- The access key for the Van Oord storage account is extracted from the Van Oord key vault.

At this point all information to setup a new model run is available in de pipeline. In the next steps the Kubernetes cluster will be prepared to run the model. Ideally the cluster will run in a completely independent Microsoft Azure account. Unfortunately, this is not a service provided by Microsoft Azure, so for this moment it is created in a separated resource group in the Deltares account.

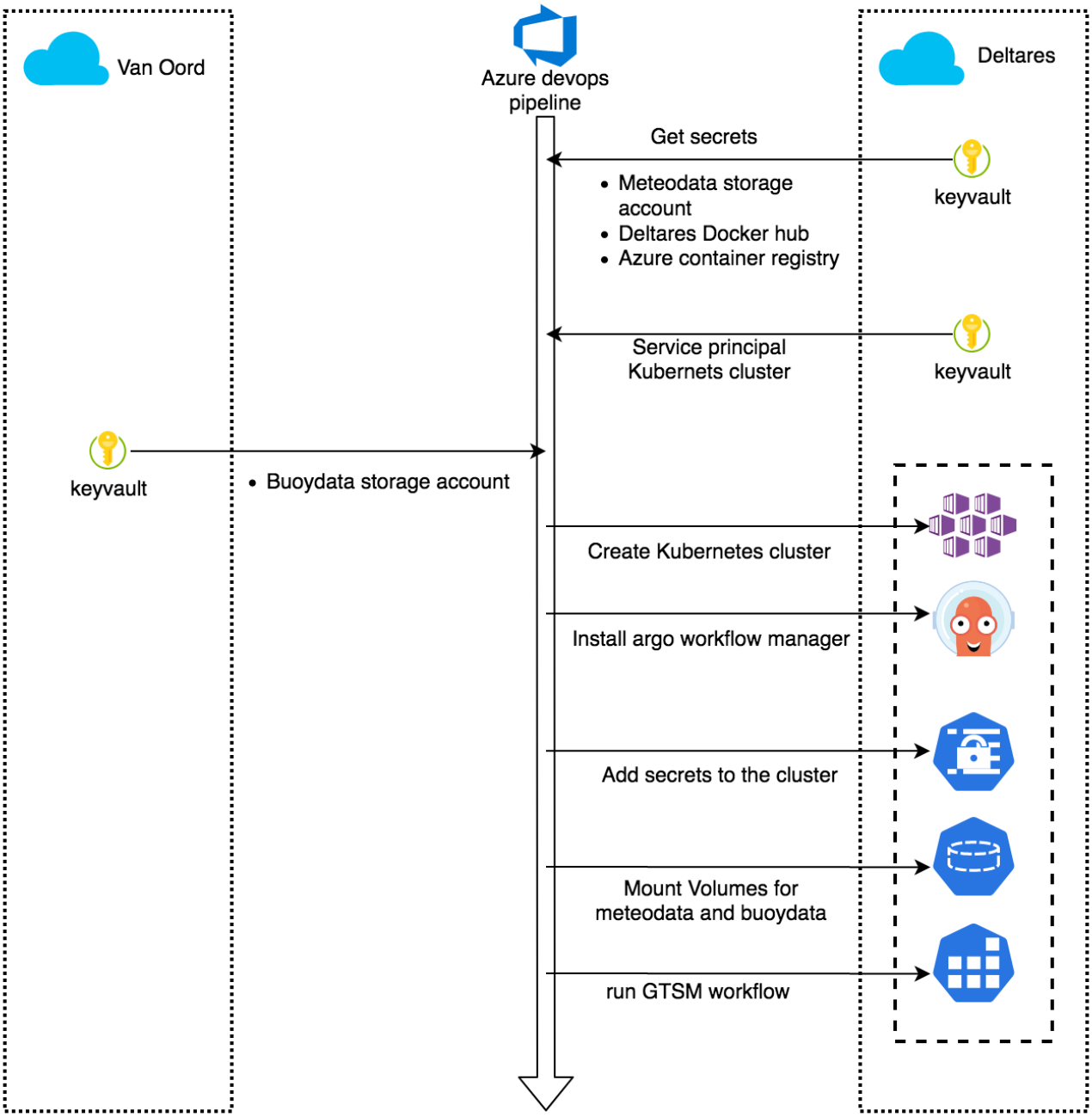
For the cluster setup we need the following steps:

- Create Azure Kubernetes Cluster with the following configuration options:

- Number of nodes
- Node size

- Install Argo workflow manager. Argo allows is to run a workflow with sequential and parallel steps.
- Install secrets. The secrets for the storage accounts and container registries are installed in the cluster.

- Mount volumes. For the storage accounts Persistent Volumes and Persistent Volume Claims are created to mount these volumes in the containers
- In the last step the Argo workflow is submitted to run the Global Tide and Surge Model.



Operation

Once the above architecture is setup, any user can trigger the workflow by making changes to the code, for example to point to a new input path or timeframe. The CI will trigger and run the complete pipeline, both to setup the architecture as to run the workflow. Larger changes to the infrastructure can be reviewed in a merge request like manner.

The last step of the pipeline is to generate a comparison between the model and the observational data, in our case between Buoy data and our GTSM model. Such a result is seen in Figure 2. Note that the output has to strike a balance between being detailed enough to improve the model and coarse enough that the original input cannot be retrieved anymore.

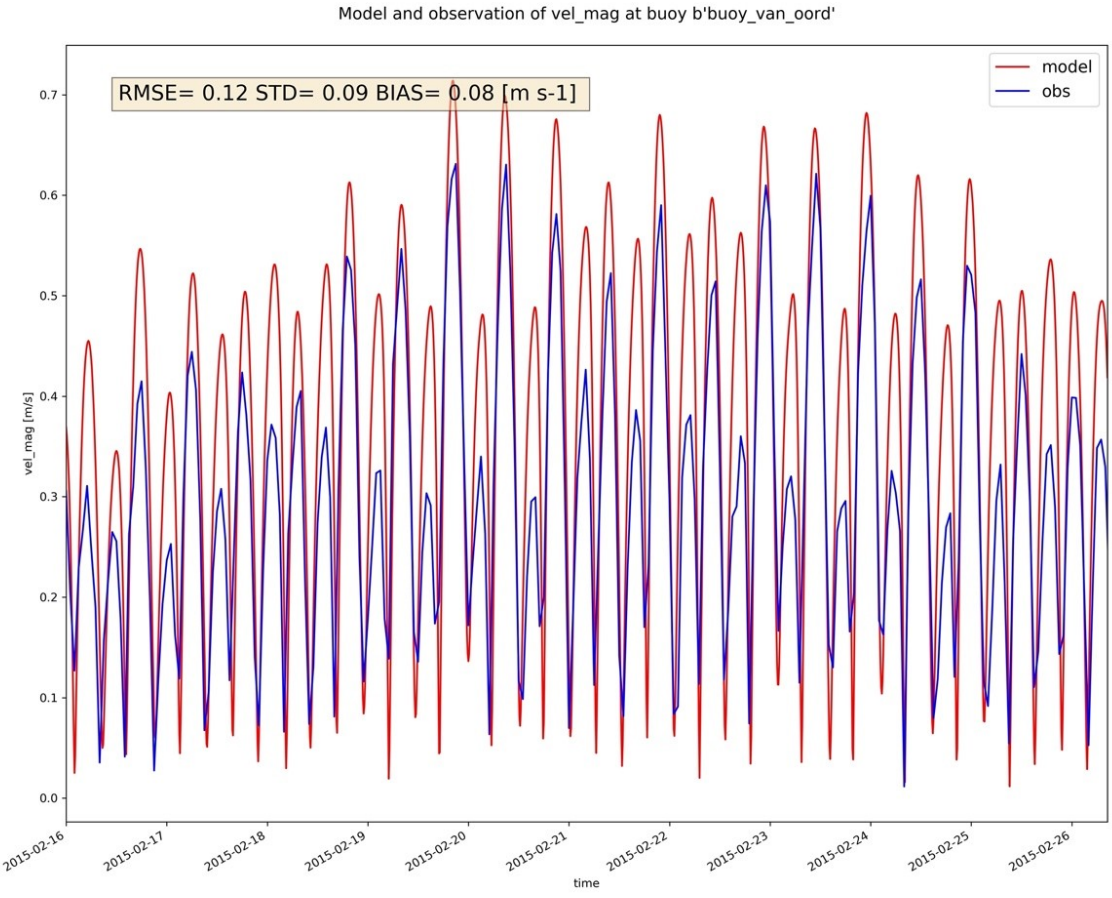


Figure 2 Output of the complete processing pipeline. The model is overestimating the magnitude of the velocity in this case, especially on lower velocities.

Future work

The above architecture and usage are a working proof of concept. We must note however that while the use of DevOps as a CI service is useful for the transparent setup of the architecture, it's less so for actually triggering the pipeline, for which a standalone service should be made. This service can however be provisioned by the above pipeline.

This concept also cannot guarantee a complete inability to access the model or data by the other party when only two parties are involved, as the service account under which the pipeline runs is still owned by of them. We also couldn't guarantee that such access, while non-trivial, would show up in logs visible to the other party.

We'll keep in touch with Microsoft to keep track of possible solutions for the above problems, as they do develop similar services for secure computing. The code for the above pipeline and preparations in Azure can be found online at https://github.com/openearth/tki-cloud-notary