Cloud-based VR
A manual for setting up a Hubs VR cloud service

Markus Stoffer

University of Amsterdam

4D Research Lab

The 4D Research Lab Report Series has been established as an instrument to promote transparency regarding the 4DRL virtual visualisation projects, workflow and pipeline development and technical experiments. The aim is to maximize knowledge sharing, meta- and paradata communication and clarification of author- and ownership of 4D Research Lab products.

Jitte Waagen, Tijm Lanjouw

Cloud-based VR – A manual for setting up a Hubs VR cloud servic

4D Research Lab Report Series 14

Author(s): Markus Stoffer
Layout: Mikko Kriek
Editor(s): Jitte Waagen, Tijm Lanjouw

All images by the 4D Research Lab unless otherwise noted.

ISSN 2772-7734

October 2025

http://4dresearchlab.nl

Introduction

This text is intended as a manual that provides instructions to set up a Hubs cloud service, which is an online collaborative Virtual Reality (VR) environment, where users can create and use online 3D virtual environments in which there can be multiplayer interaction and collaboration. Funded by the Incentive scheme Open and online education[1], this manual is produced in the context of the Virtual Past Places project (Waagen et al. 2024). For this project, the 4D Research Lab explored the potential of Hubs for educational purposes, by designing multiple virtual reality learning environments together with teachers and technology-enhanced learning experts. A Virtual Past Places webpage has been launched in context of this project (https://www.virtualpastplaces.eu/).

VR technology is a relatively inaccessible technology for several reasons. Firstly, VR software, e.g. VR learning applications, is often exclusive to particular hardware or app stores, often related to specific compatibility of apps. Secondly, not every school or university is in possession of VR headsets and the expertise to do basic set-up and installation. Finally, documentation of VR development is regularly unorganized and/or very specific, due to the multitude of systems and system versions, and due to the niche expertise of developing and customizing a virtual reality app.

Hubs tackles some of these issues by being a VR creation and user platform available on both web browsers and headsets. However, the decommissioning of its original servers left users to fend for themselves in terms of server connection. To approach the sparsely documented process of setting up the service, this manual was produced to enable potential users to make and host their own Hubs cloud service, and with videos linked in the footnotes, also shed light on the processes of creating 3D environments and preparing them for interactive collaboration.

[1] Incentive Scheme Open and Online Education. https://web.archive.org/web/20241103133604/https://www.surf.nl/en/education-it/incentive-scheme-open-and-online-education (accessed 2 September 2025).

1 Context of this manual

1.1 A short history of Hubs

Our use of the technology of Hubs, previously Mozilla Hubs, has progressed alongside the turbulent recent history of its hosting and development. From the moment we started using this, it was easy to set up Hubs using the service via the Amazon Web Services (AWS) Marketplace. The entire hosting side was prepared beforehand and offered as a service using the Marketplace. This was a workable and consistent solution, although with the limitations of only being able to use AWS. In 2023, Mozilla announced the decision to replace the self-hosted (by then highly out of date) Hubs Cloud with a 'Community Edition', meaning the user would decide how and where to host the service. Initially, this was a decision to enable users to choose other services besides Amazon. By the end of 2023, a subscription service hosted by Mozilla was launched to keep the option of outsourcing the server-side maintenance, but by the beginning of 2024 it was announced that Mozilla was discontinuing the entire Hubs service, and that users had three months to download their data. Unfortunately, documentation and tools for this were only available in the last weeks of this period. Currently, accessing any of the previous Hubs documentation pages or resources now opens an 'end of support' page.[2]

These events posed unexpected challenges to the project and required work on the server-side to be done by ourselves. Furthermore, many times when a change in service occurred, our scenes and data had to be downloaded and reuploaded, causing further delays. In some cases, this caused loss of data because of a lack of backups or persistent databases. This text documents the process to easily set up Hubs on DigitalOcean, so that subsequent users are able to make their own Hubs, thereby also facilitating the possibility of using the VR environments created in the context of the Virtual Past Places project.

1.2 Setting up a Hubs Cloud with DigitalOcean

This subchapter will describe how to set up a Hubs Cloud instance using the service provided by DigitalOcean. This cloud service provider has multiple available services, but what makes it a good fit for Hubs is its Kubernetes management service, the software deployment system Hubs utilizes. Kubernetes is a system that automates and manages the different parts of Hubs, and is facilitated by many cloud services, including DigitalOcean. This is notably more affordable than other commonly used services, such as Google Cloud Platform (GCP) or Amazon's AWS (table 1).

---

[2] End of Support for Mozilla Hubs. https://support.mozilla.org/en-US/kb/end-support-mozilla-hubs/ (accessed 2 September 2025).

Platform	Cluster	Price
GCP	Custom machine, 2 vCPUs, 4GB RAM, series 'N1'. Region Netherlands (europe-west4)	€135.18
AWS	Single cluster, standard support. Unclear machine specifications. Region Europe (Frankfurt)	€67.47*
DigitalOcean	Shared CPU, 2 vCPUs, 4GB RAM, 80GB storage. Amsterdam – Datacenter 3.	€22.18*

Table 1: a comparison of costs of three different Kubernetes-hosting services. Costs with an asterisk have been converted from US dollars to euros, with the exchange rate of 06-03-2025.

Notably, GCP and AWS both have options to add a so-called 'persistent volume', a piece of storage reserved for long-term data storing, but when hosting using DigitalOcean, choosing a backup option along with backing up scenes and materials offline are essential to prevent loss of data.

1.3 Layout of different components

The following setup is applied and followed throughout this manual (fig. 1). To edit the Hubs codebase hosted on GitHub, use a code software such as Visual Studio Code. Git is also useful for users familiar with it. Install Node.js to enable required command line tools, and download the Kubernetes-specific tools 'kubectl' and 'doctl' to deploy and connect to the cluster on DigitalOcean. Three components must be prepared online: you need a registered domain name for accessing the website, you need a Simple Mail Transfer Protocol (SMTP) server for sending invitation links, and you need a server with Kubernetes compatibility. In this guide, the domain name and SMTP server are hosted on AWS, while the Kubernetes cluster is deployed on DigitalOcean. If necessary or useful, some of these components can be substituted with other services. For example, Porkbun[3] for the domain name and Scaleway for the SMTP server[4] are independent alternatives to AWS. For a setup that does not include AWS nor DigitalOcean, the France-based Scaleway can be used for setting up the Kubernetes cluster as well as the SMTP (mailing) server. Other viable options are OVHcloud in France, and Exoscale[5] in Switzerland.

3 Porkbun. https://porkbun.com/ (accessed 2 September 2025).

4 Scaleway - Setting up SMTP. https://www.scaleway.com/en/docs/transactional-email/reference-content/smtp-configuration/ (accessed 2 September 2025).

5 Exoscale. https://www.exoscale.com/ (accessed 2 September 2025).

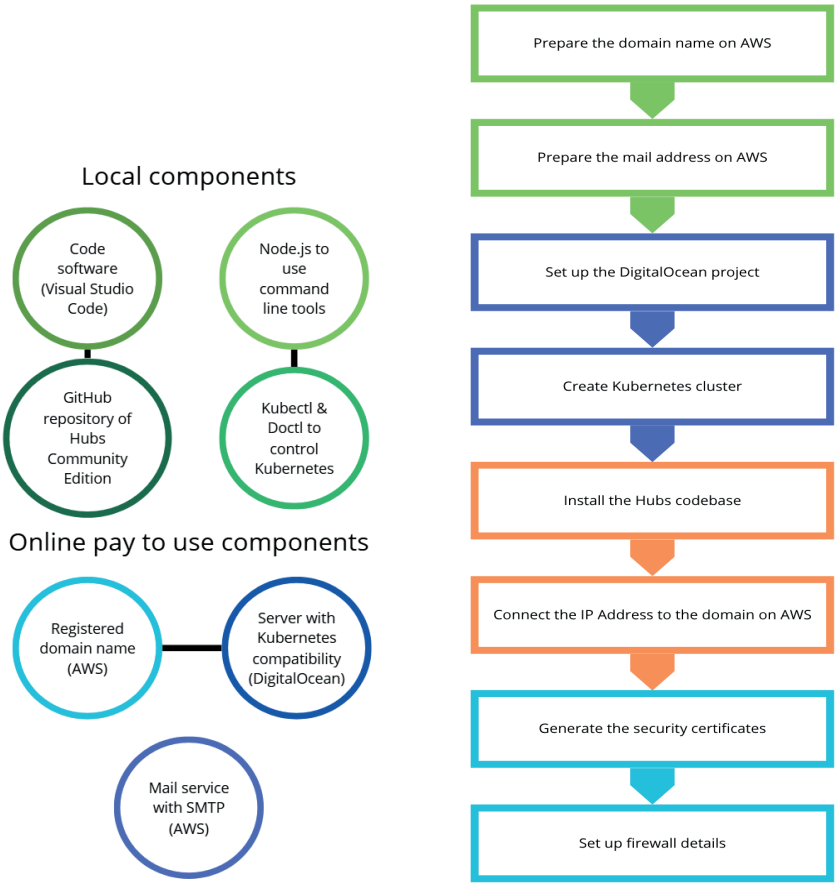


Figure 1. Left: The separate components that form the hosting process described in this manual. The linked components are as follows: the code software edits and deploys the code from GitHub, Node.js is necessary to install and use Kubectl and Doctl, and the Kubernetes cluster is deployed to the registered domain name. Right: The steps covered by this tutorial. The first part in green is the preparation on AWS, the second part in purple is the preparation on DigitalOcean, the third part in orange is the setup of the Hubs code to the point where it is reachable online, and the last part in blue contains tweaking settings so that the website is reachable for other people without any issues.

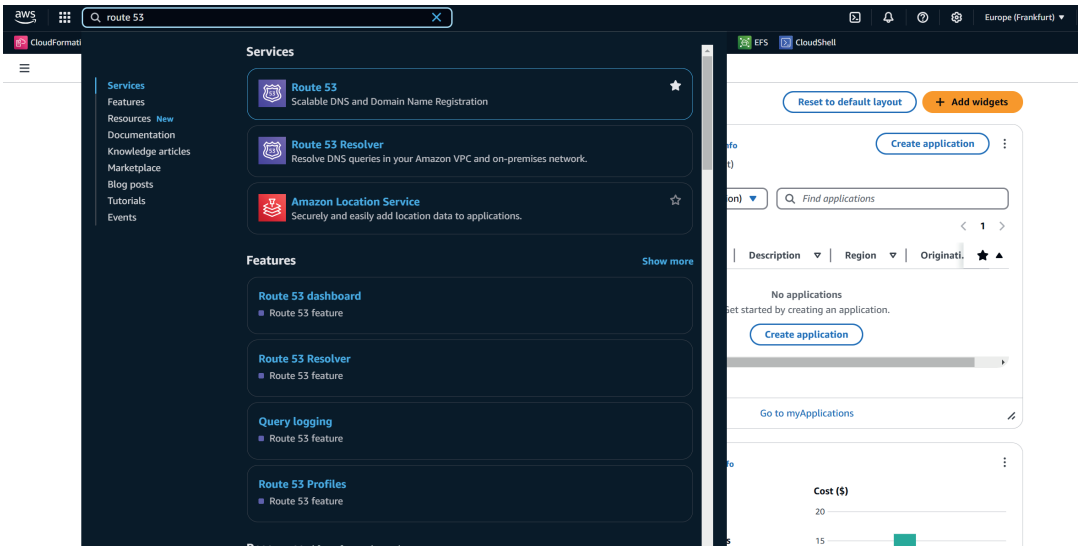


Figure 2. The AWS console showing the 'Route 53' service upon logging in and using the searchbar.

2 Step-by-step guide to set up Hubs

2.1 Preparing the domain

The first necessity is a registered domain, the reserved name and hosting of the website necessary to launch Hubs. Use Route 53, AWS' Domain Name System (DNS) service to register the domain and enable translation from website name to the IP address where the cluster is hosted. To begin, register an AWS account[6] and use the search bar to find Route 53; the service will show as the top option in the results (fig. 2). In the top-right of the screen, the 'zone' is visible in which the current AWS service is functioning. For Route 53, the zone is 'Global', as domain names are not bound to a particular region.

To register a domain, click 'Registered domains', which can be found in the sub-menu 'Domains' in the left sidebar. Click the orange button that reads 'Register domains', at the top-right of the screen. This will lead to the creation screen of Route 53 (fig. 3). Choose a top-level domain name (the part after the domain name in the URL), like .com or .eu. The costs vary depending on the choice: the popular .io domain costs $71, .com, .eu and .org cost around $13, and national ones such as .nl and .co.uk cost around $10.

In case the chosen name is still free, and you can proceed (fig. 4). Purchase the domain by clicking 'Proceed to checkout'.

Finish the checkout step, and the first preparation for setting up Hubs is done. The domain name can be accessed, but nothing is visible yet. Later on, the domain will be linked to the Hubs instance set up on DigitalOcean.

2.2 Preparing the email address

The second preparation is arranging a mailing service, necessary for sending automatic mails to users. Hubs uses this for sending invitation links to new users. In the AWS console, search for 'Simple Email Service' (fig. 5).

Go to the Simple Email Service menu, and choose the AWS region (i.e. the geographic location of the data center) in which the service will operate. The region button is visible at the top-right of the screen. Change it to the server nearest to any expected users or the server location (fig. 6). Afterwards, go to 'SMTP settings', the fourth option of the sidebar on the left. This displays information important to enter in the code for Hubs later. Write down the SMTP Endpoint (called SMTP_Server by the Hubs code) and note that the port used is 2587 (this is a port both AWS and DigitalOcean accept, as traffic on the common port 587 is automatically blocked). Then click 'Create SMTP Credentials' on the top right.

---

6 How to create an AWS account. https://aws.amazon.com/resources/create-account/ (accessed 2 September 2025).

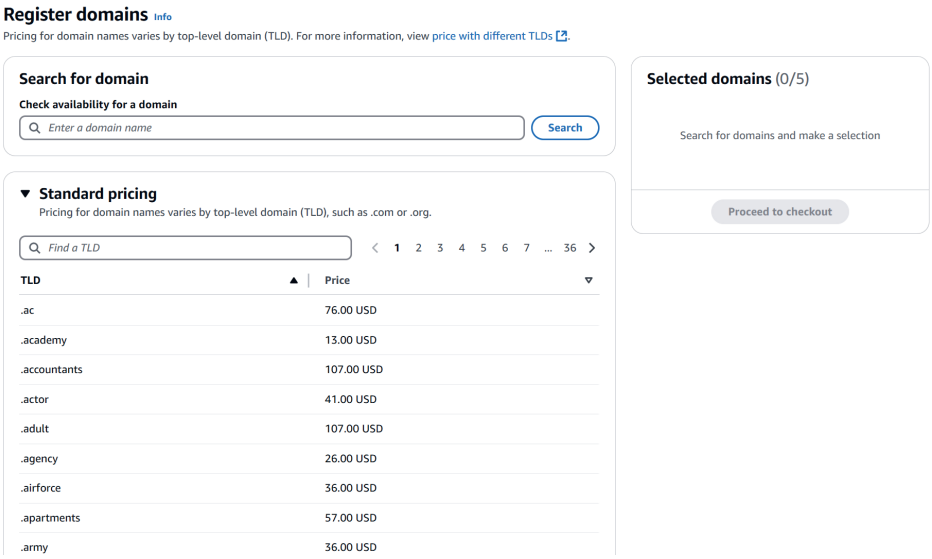


Figure 3. The 'Register domains' menu with the pricing for a few top-level domain names

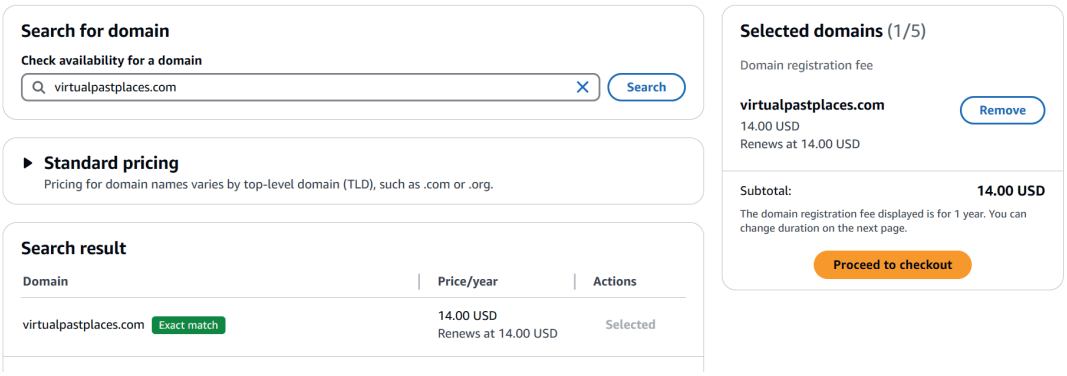


Figure 4. The exact match to the searched domain name is free and is added to the 'selected domains' to be purchased.

This displays a new console menu, the IAM (Identity & Access Management) console. Choose a name and create, then write down the "SMTP User Name" (different from the IAM user name) and the "SMTP Password".

Go back and open 'Identities' under 'Configuration', and create two new identities, one domain and one email address. Click 'Create Identity' and choose domain, then fill in the domain name in the format "virtualpastplaces.com". Default settings for the other fields are alright. Finish, and create another identity, choosing the email address type. Fill in your intended admin email address, and verify using the mail you will receive. Once both identities show 'identity status: verified' in the 'Identities' screen, the preparation is finished. These are all the things needed from AWS, so continue by setting up DigitalOcean.

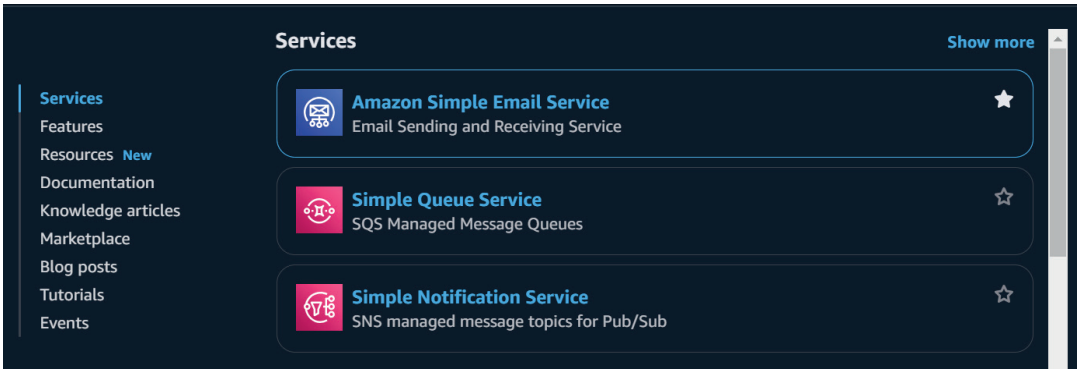


Figure 5. The AWS console showing the 'Simple Email Service' upon using the searchbar.

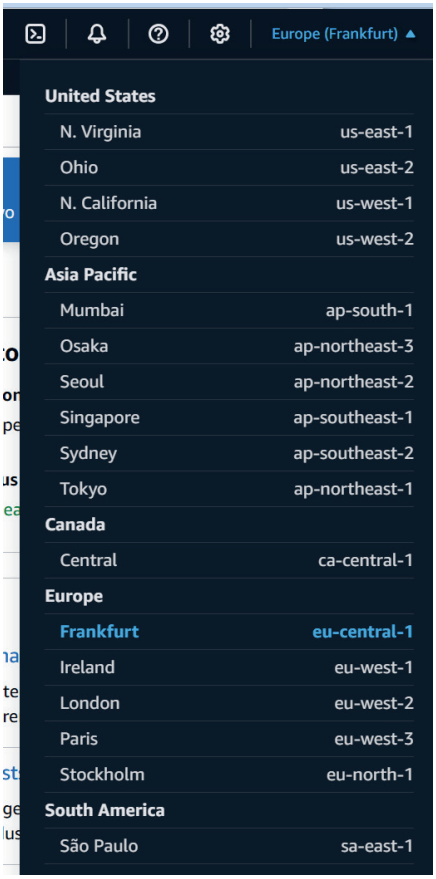


Figure 6. A preview of possible regions upon clicking the region button at the top-right of the AWS console.

2.3 Starting up with DigitalOcean

Go to the DigitalOcean website[7] and click 'Sign up' on the top right. Signing up can include email verification, basic demographic feedback, and a payment method verification. After setting up the account, the starting screen is visible, with a default 'first-project' already prepared (fig. 7).

[7] DigitalOcean. https://www.digitalocean.com/ (accessed 2 September 2025).

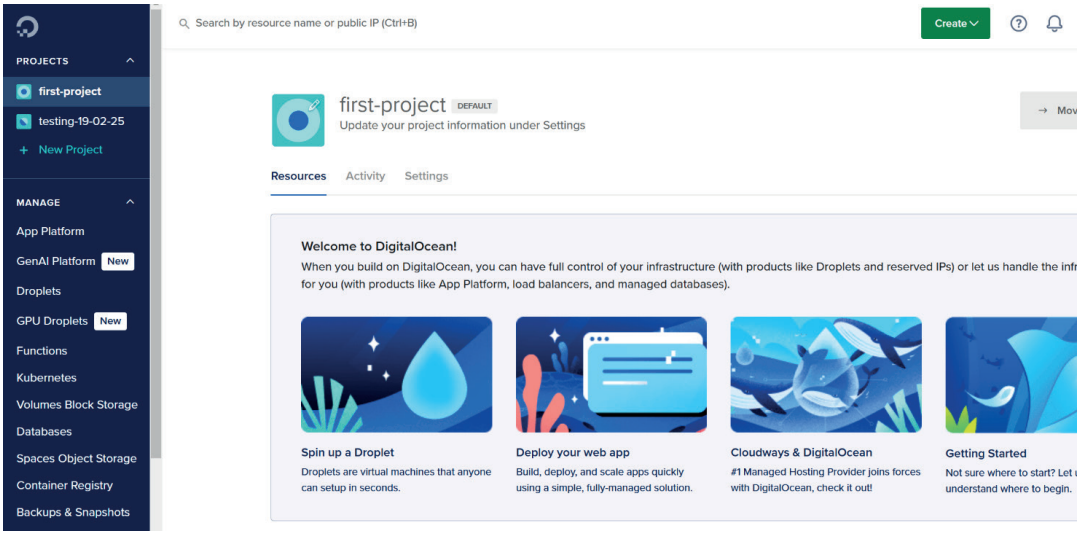


Figure 7. The DigitalOcean console upon logging in, showing the default 'first-project'.

2.4 Creating the Kubernetes cluster

To start creating the Kubernetes cluster necessary for Hubs to run, click the sidebar option reading 'Kubernetes', under 'manage'. Then click 'Create a Kubernetes Cluster' to get started. In the screen with cluster options (fig. 8), select the recommended version. For 'datacenter region', pick the location closest to the server location and expected audience. The 'VPC Network' doesn't have to be changed.

For cluster capacity (fig. 9), the following settings are recommended based on testing: give the node pool a recognizable name, then choose Shared CPU, machine type 'Basic', with a node plan of 24$/month without autoscale, with a single node. In practice, single-node clusters were observed to be more stable than autoscaling clusters on DigitalOcean. Note that Hubs cannot function on a cluster lower than 4 GB RAM, making this the cheapest viable option.

Give the cluster a name and optionally add tags that describe the intended purpose of your project. Finally, click 'create cluster'. Then, to continue with connecting to the cluster, the command line tools for Kubernetes are required. Kubectl can be obtained via the following link for windows: https://kubernetes.io/docs/tasks/tools/install-kubectl-windows/ or by using the options in the side menu for other operating systems. To check if kubectl has been installed correctly, use the command prompt (type 'command prompt' in the windows searchbar to open), and copy and paste the following command:

kubectl version

If this command returns version information (for example a Client version and a Kustomize version), kubectl is installed correctly. Next, 'Doctl' is required – this is an equivalent command line tool used for specifically managing DigitalOcean-based Kubernetes clusters. To install it, follow the documentation on the

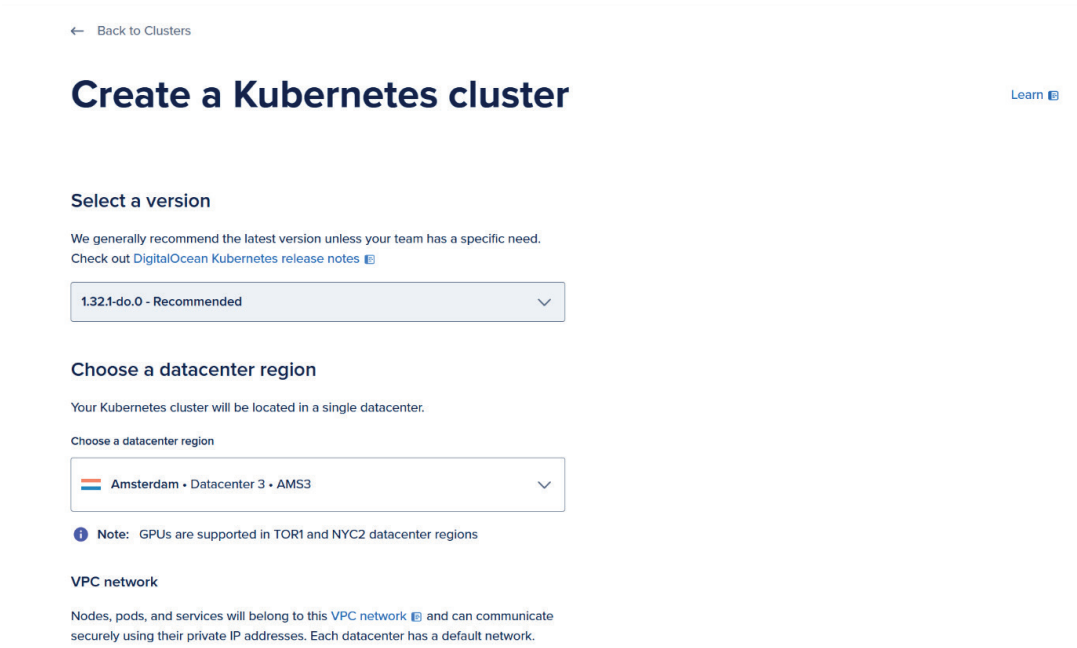


Figure 8. The first few options visible when creating a Kubernetes Cluster on DigitalOcean.

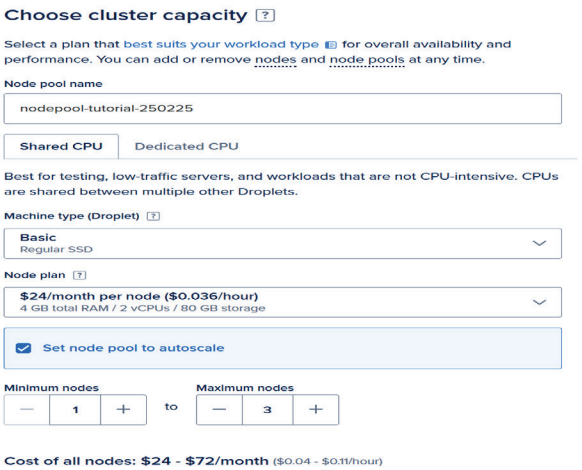


Figure 9. The options for cluster capacity visible when creating a Kubernetes Cluster on DigitalOcean.

DigitalOcean (https://docs.digitalocean.com/reference/doctl/how-to/install/), and visit the GitHub page with doctl releases (https://github.com/digitalocean/doctl/releases). download the one appropriate for your system (for example, Windows-AMD64). Extract the downloaded .zip file, then place the extracted files into a folder with a short path, such as C:\code. After this, type 'environment variables' in the windows search bar, and open the corresponding system settings screen (fig. 10). This is necessary to manually make doctl accessible using the Windows command line by adding its file path.

In this screen, click the button at the bottom reading 'Environment Variables...', then under 'User variables for user', select the variable named 'path' and click

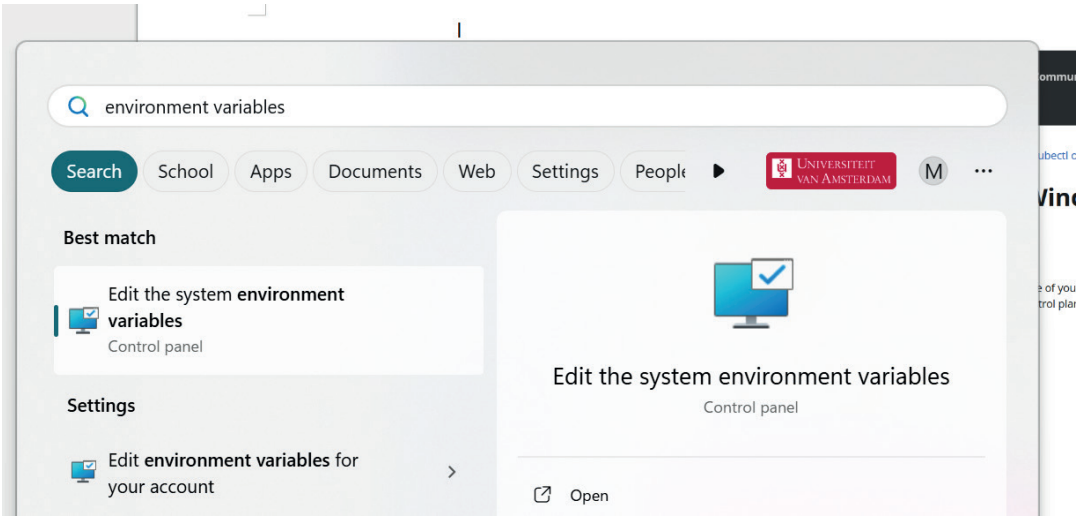


Figure 10:. Finding the systems setting for editing the system environment variables using the Windows Searchbar.

'edit'. Then click "new", and paste the path to the folder where the doctl executable is located (for example, 'C:\code'). Confirm the changes by clicking 'OK'. Afterwards, the configuration of doctl can be verified by executing the following command in the command prompt:

doctl version

The expected response is a doctl version release number, and a Git commit hash (an identifier for Git). This means doctl is ready to be used. Go to DigitalOcean and scroll all the way down in the sidebar and click "API", one of the options at the bottom (fig. 11). Here an API key can be created to securely connect the computer with the Hubs files to the cluster on DigitalOcean.

Click 'Generate New Token', give it a name, select 'No expire' for the expiration date, 'Full Access' for scope, then scroll to the bottom and click 'Generate Token'. Once complete, save the 'secret' (the private token needed to connect), as it is required in a later phase and will not be shown again. Now, using command prompt, use doctl to connect to DigitalOcean. Copy and paste the following line, in this case using 'HubsContext' as the name for the context:

doctl auth init --context HubsContext

The console will ask for the DigitalOcean API token. Once it is entered and has been validated, copy and paste the following line to switch the active context to the newly made one:

doctl auth switch --context HubsContext

The command line response will read: 'Now using context [HubsContext] by default'. Go back to the Kubernetes page and select the cluster, then go to the

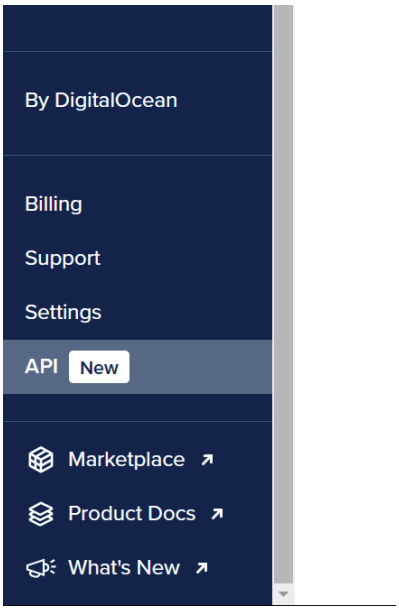


Figure 11. The sidebar of the DigitalOcean console, showing the 'API' option.

second step, 'Connecting to Kubernetes'. Copy the line that is used to authenticate and paste it into the command line screen. Two notifications appear to confirm the context of the cluster was successfully added. Optionally verify connectivity using the commands shown in step 3, and skip step 4 (this step is for setting up sample apps, whereas Hubs can directly be installed to the cluster). Click 'Great, I'm done' to finish the installation.

2.5 Installing Hubs

The code for Hubs is maintained by the Hubs Foundation and can be found on their GitHub (https://github.com/Hubs-Foundation/hubs-cloud). Download it by clicking the green 'code' button on the top-right and selecting 'Download ZIP' (fig. 12). Extract the files and put these in a location with a short file path, like C:\HubsCE. Open Visual Studio, and in the new folder, open the hubs-cloud-master folder.

In the folder 'community-edition', open the input-values.yaml (see figure 13). Change "HUB_DOMAIN" to the domain from AWS using the format 'virtualpastplaces.com', and change "ADM_EMAIL" to a preferred email address to be listed as the first administrator of the Hubs instance, using the format 'name@email.com'. Go down to line 13 and change SMTP_SERVER to the Endpoint of AWS written down when preparing the email address, change SMTP_PORT to '2587', and change SMTP_USER and SMTP_PASS to the SMTP username and SMTP password. Changing this file also allows the optional use of SketchFab and Tenor with Hubs. To prepare this, search for "API token" in combination with the respective service and paste the token in the respective field. Change the node, guardian and phoenix attributes to a random string of numbers and lowercase letters, and do not change these after the initial setup. Save the file and find the folder in windows explorer. When in the community-edition folder, click on the data path, and type 'cmd' to open the command prompt in this exact location (fig. 14).

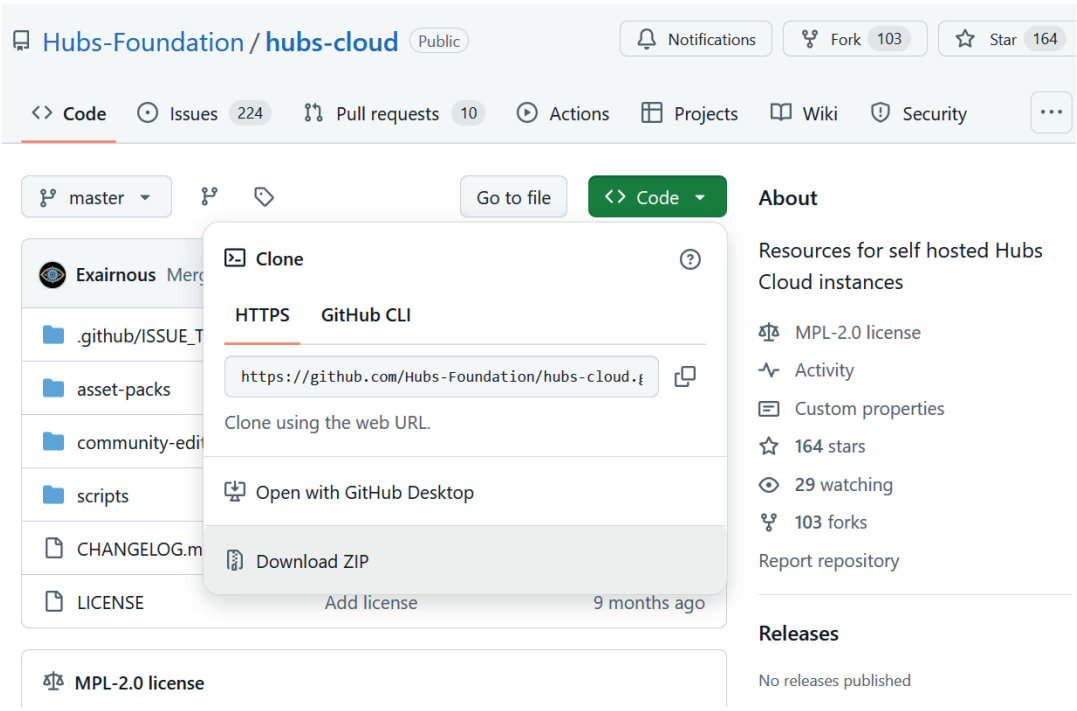


Figure 12. The page where it is possible to download the Hubs Community files on the GitHub page of the Hubs Foundation.

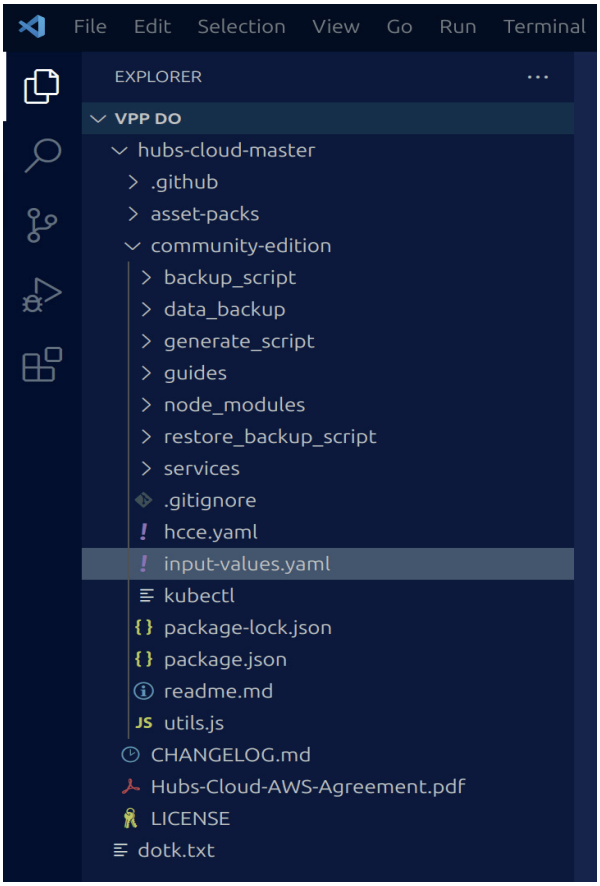


Figure 13. The file explorer of Visual Studio Code with the 'hubs-cloud-master' folder opened and the input-values.yaml file highlighted.

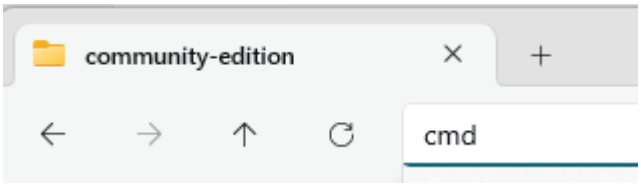


Figure 14. Typing 'cmd' in the Windows Explorer searchbar to use the shortcut to command prompt.

In the command prompt, run the following command to install the dependencies (referenced necessary files and plugins in the Hubs code):

npm ci

The expected output is "added n packages, and audited n packages in n s(econds), found n vulnerabilities". Afterwards, run:

npm run gen-hcce

This will produce a 'hcce.yaml' file, a configuration file based on the values filled into the input-values file. Installing and preparing the Hubs files is now complete, so continue to bring the cluster with the Hubs instance online.

2.6 Bringing the cluster online and connecting to AWS

With the prepared hcce.yaml file, it is time to deploy to the kubernetes cluster. Open the command prompt, and if not currently in the community-edition folder, use the following command to move there, with the path name being the location of your Hubs code:

cd C:/hubs-cloud-master/community-edition/

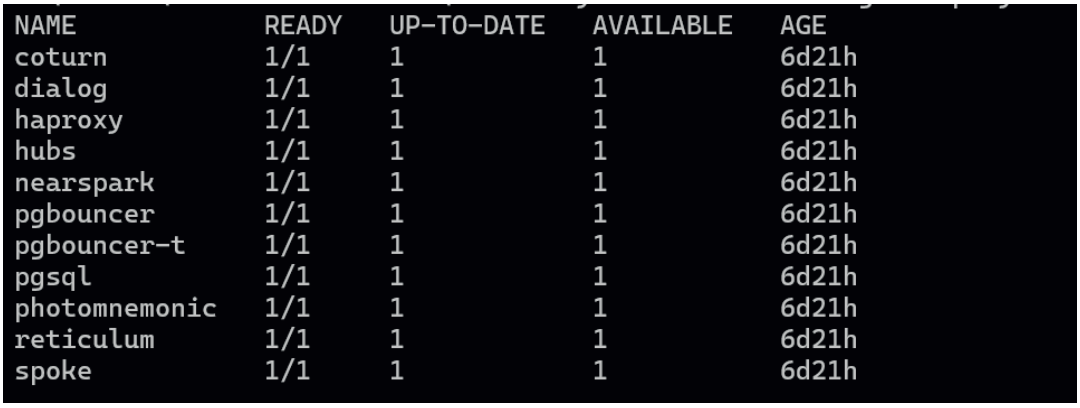
While in the community-edition folder, paste the following command into the command prompt, to apply the prepared configuration to the Kubernetes cluster on DigitalOcean.

kubectl apply -f hcce.yaml

The feedback will show multiple things being created. When the task has finished, the command prompt will show the file path again. To verify the deployed cluster, paste the following command:

kubectl get deployment -n hcce

This will give a result like in figure 15, showing the separate pods of the Kubernetes cluster coming online.



NAME	READY	UP-TO-DATE	AVAILABLE	AGE
coturn	1/1	1	1	6d21h
dialog	1/1	1	1	6d21h
haproxy	1/1	1	1	6d21h
hubs	1/1	1	1	6d21h
nearspark	1/1	1	1	6d21h
pgbouncer	1/1	1	1	6d21h
pgbouncer-t	1/1	1	1	6d21h
pgsql	1/1	1	1	6d21h
photomnemonic	1/1	1	1	6d21h
reticulum	1/1	1	1	6d21h
spoke	1/1	1	1	6d21h

Figure 15. The kubernetes deployment after it came online completely.

After a few minutes all pods will display “1/1” under the “ready” header. This means that the separate components of the cluster have completed starting up without errors. Afterwards, the external IP address is required to link it to the domain name. This ensures that the purchased domain directs to the cluster on DigitalOcean. Paste the following command in the terminal to find the ‘load balancer’, the component which shows the external IP:

kubectl -n hcce get svc lb

The IP address shown under ”External-IP” is the address where Hubs can now be reached on the web. Copy this address, so it can be used in the Route 53 service. Since Route 53 hosts the domain name, it needs the correct information so that Hubs on DigitalOcean can connect to the domain, and becomes reachable by visiting the website. Go to the AWS console, to Route 53 (using the search bar or the ’recently visited’ screen on the top left), and click ’Hosted Zones’ in the left sidebar. Click the domain here, so that the records are visible. Click ’create record’ to open a screen where it is possible to fill in values for new records. Create four new records (this can be done in a single ‘create’ action), all with the same record type (A), TTL (300) and Routing Policy (simple routing). These attributes mean that the created route is a regular link from the domain to an IP (IPv4) address, with 300 seconds delay before information is updated (a security measure to prevent unwanted sudden changes). In ”Value”, fill in the IP copied from the ‘External-IP’ of the load balancer. For the first record, the record name is empty, meaning the resulting URL is simply .domainname, without a prefix. Click ”Add another record” and make an almost identical record, including the external IP, except with the record name being ’assets’. Do this two more times, one record name being ’stream’ and one record name being ’cors’. This creates a total of four new records, which allow the domain to direct to the Hubs instance and its related data.

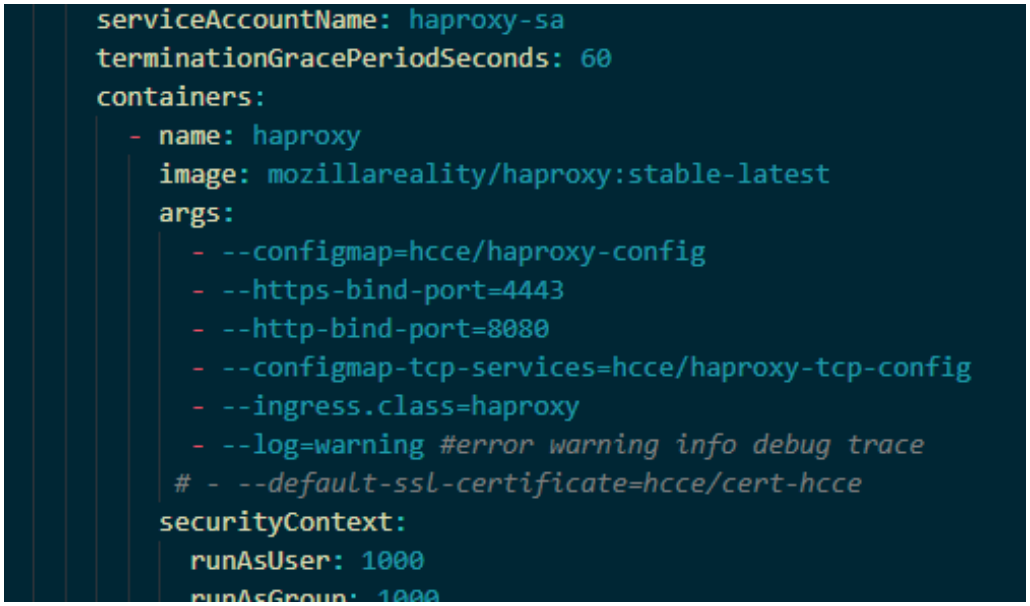


Figure 16. The hcce.yaml changed to not make use of the default ssl certificate.

2.7 Setting up security certificates

After a while, the domain can be reached, but for now shows a security warning, caused by a lack of valid security certificates. To generate these, use a npm function included in the files downloaded from the Hubs Github. This function generates the necessary security certificates, with a validity of 90 days. Go to the command prompt, and paste the following command:

npm run gen-ssl

Check that in the resulting output, there are four certificates, saying cert-, cert-assets, cert-stream and cert-cors. If one of them is still missing, run the command again. By default, a placeholder certificate is present, which needs to be replaced by the certificates that were just generated. In order to turn off the default certificate, go to the hcce.yaml file and go to line 1315, or use the control plus ‘F’ command to search for ‘default-ssl-certificate’. ut a hash/number sign in front of this entire line (fig. 16). Save the file, and once more run the apply command:

kubectl apply -f hcce.yaml

Afterwards, the Hubs instance is accessible using the domain name. To prevent browsers from storing the default certificate, which causes the security warning to keep appearing, it may be necessary to clear the browser cache or try the URL on a different browser.

Now that it is possible to enter Hubs, use the Sign in/Sign up button and fill in the admin email address, and use the link in the email to access Hubs’ administrator panel. The ’Admin’ option will be visible in the main screen (fig. 17).

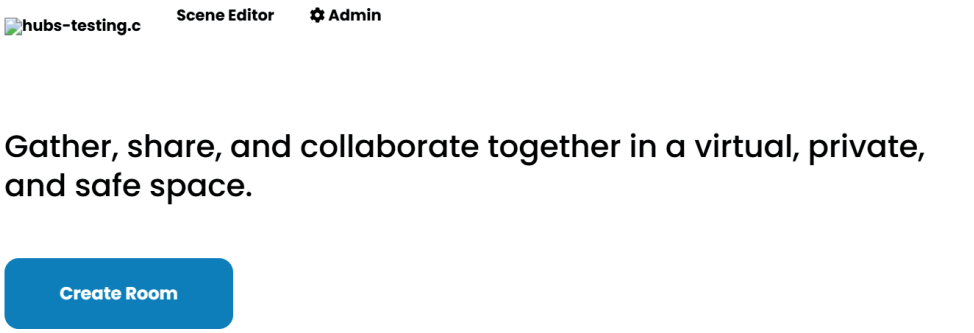


Figure 17. The Hubs homepage, showing the link to go to the Admin menu. For now, the broken images are not a problem, these are added by the admin to personalize the page

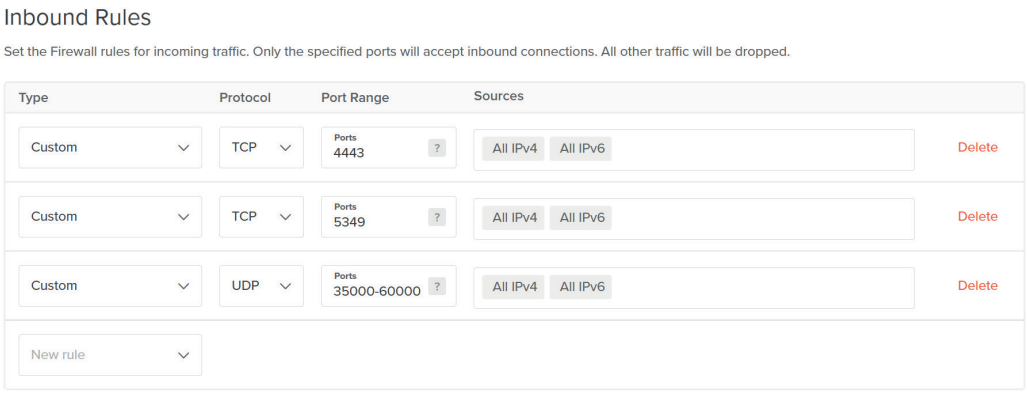


Figure 18. The inbound rules of the firewall necessary for DigitalOcean.

2.8 Setting up firewalls on DigitalOcean

For its interactive multiplayer functionality, Hubs also requires certain firewalls to only allow intended traffic. To do this, go to the DigitalOcean console, click 'Networking' in the left sidebar, and select 'Firewalls'. Click 'Create Firewall' and make three Inbound Rules: set all to the 'custom' type, set one protocol to TCP and port 4443, set one protocol to TCP and port 5349, and the third protocol to UDP and port range 35000-60000 (fig. 18). Where it says 'Apply to Droplets', enter the name of the node pool: typing the first letter shows the node pool of the cluster to appear as the suggested result. When applied, Hubs is now fully operational.

Further setting up

To continue setting up Hubs, five tutorial videos have been produced and posted on the YouTube Channel of the 4D Research Lab (https://www.youtube.com/@4dresearchlab97). These reproduce and explain the utilized workflow in fit for first-time users. The two videos produced in 2023 consider the user's side, one covering the interaction and navigation when a user enters a Hubs scene (https://www.youtube.com/watch?v=2STIB_86nhk), and one covering object placement and interaction with 3D objects(https://www.youtube.com/watch?v=nxv5uwobUL8) In 2024, three tutorials where added that cover the creator's side, one setting up the 'backend' for hubs using the Google Cloud Platform (https://www.youtube.com/watch?v=AnPcsXQKAsA, with a partial overlap with the instructions in this text), one showing the scene editor(https://www.youtube.com/watch?v=Xvt8dYjQKWchttps://www.youtube.com/watch?v=Xvt8dYjQKWc), and finally one showing the creation of a 'virtual exhibition' with a set of functions to show and describe images and objects in online 3D space (https://www.youtube.com/watch?v=jFYdBc08sY0).

Acknowledgments

This manual was written by developer Markus Stoffer. The manual builds also on the expertise of the Virtual Past Places project lead and coordinator Jitte Waagen, data engineer Kwan Suppaiboonsuk and frontend developer Matúš Solčány, and author of the Hubs Foundation's documentation, immersive learning expert Heather Dodds. The Hubs tutorial videos linked in the references were produced by 3D technician Tijm Lanjouw and the author of this report. Further acknowledgments specific to Virtual Past Places can be found on the 4D Research Lab Website (https://4dresearchlab.nl/virtual-past-places/).

References

- Waagen, J., Lanjouw, T., Huurdeman, H., 2024. Virtual past places. Reinventing the classroom using virtual reality in humanities education. AHM Conference 2024: "*Heritage, Memory and Material Culture*". 78–84. Amsterdam University Press. https://doi.org/10.5117/9789048567638/ahm.2024.014.