Unreal Horde Server Deployment (Docker)

Hidde Derks

02 Dec 2025 — 12 min read

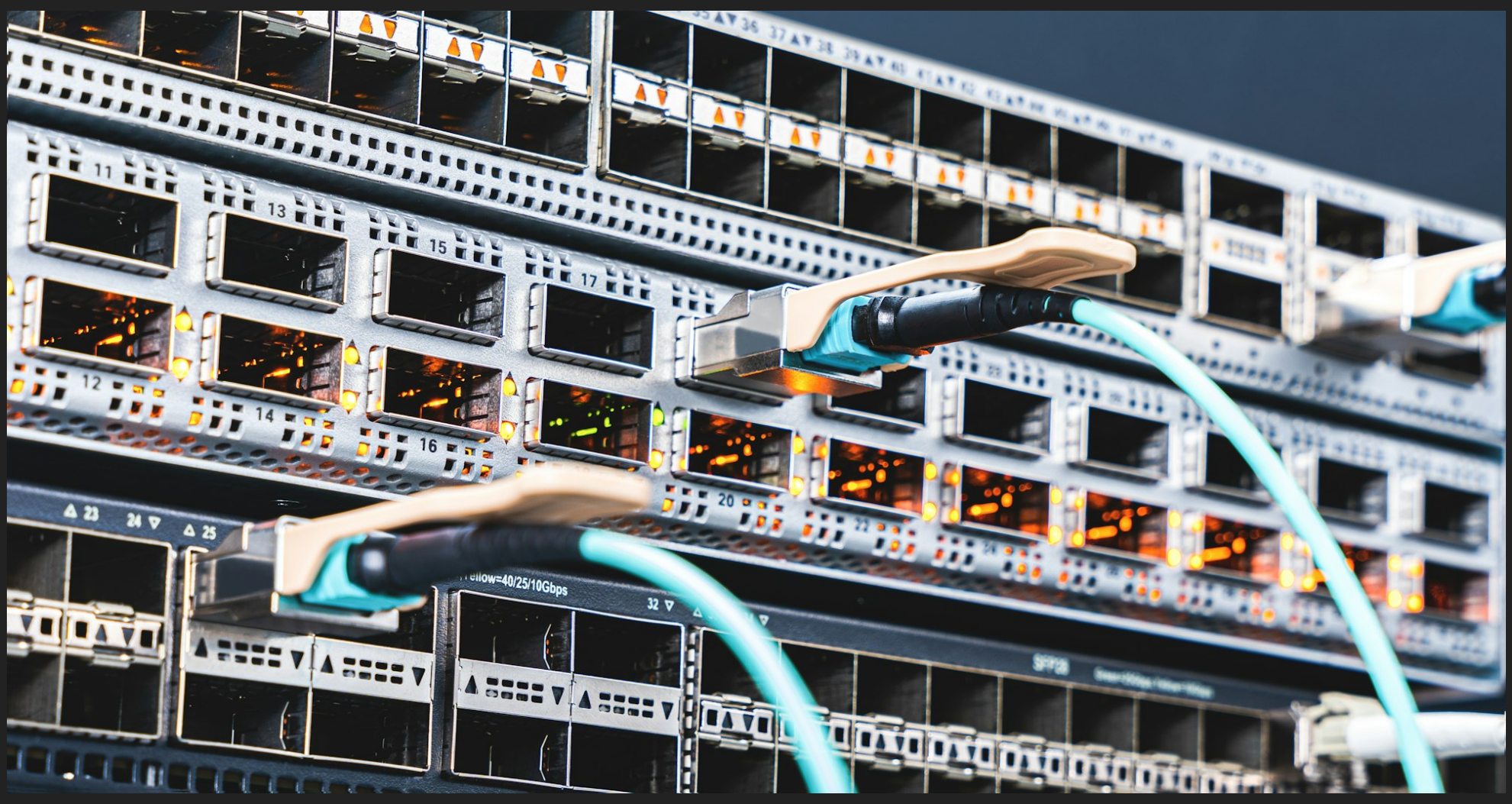


Photo by Lightsaber Collection / Unsplash

Step 1.

Step 2.

Step 3.

Step 4.

Step 5.

Step 6.

Step 7.

Step 8.

> Initially created for the Breda University of Applied Sciences Master's of Game Technologies

To start benefiting from the Unreal Build Accelerator you need to have a server set up for agents to connect to. In case you're looking for agent set up, choose one of the following:

Horde Agent (Windows)

Horde Agent (Docker/Linux)

This guide will focus on deploying the Horde Server in Docker, if you're looking for the Windows deployment, go here:

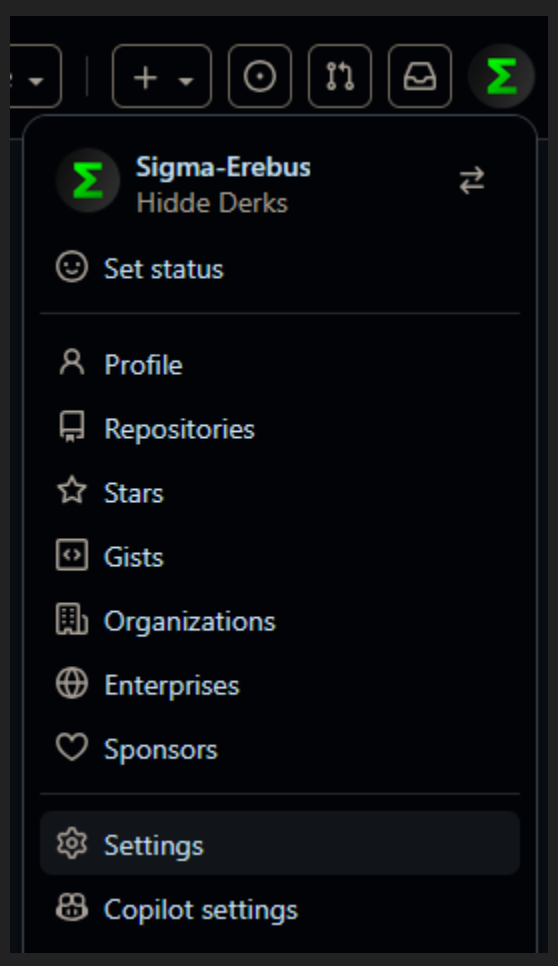
Horde Server (Windows)

Before getting started with this guide, you do need to have access to Unreal Engine's source code on GitHub, guide for which can be found here: https://www.unrealengine.com/en-US/ue-on-github. And a machine capable of running Docker that will become your Horde Server.
This can be a Linux server or other containerization backend. Docker Desktop can also be useful as a GUI, or as debugging tool, but will not be a requirement.

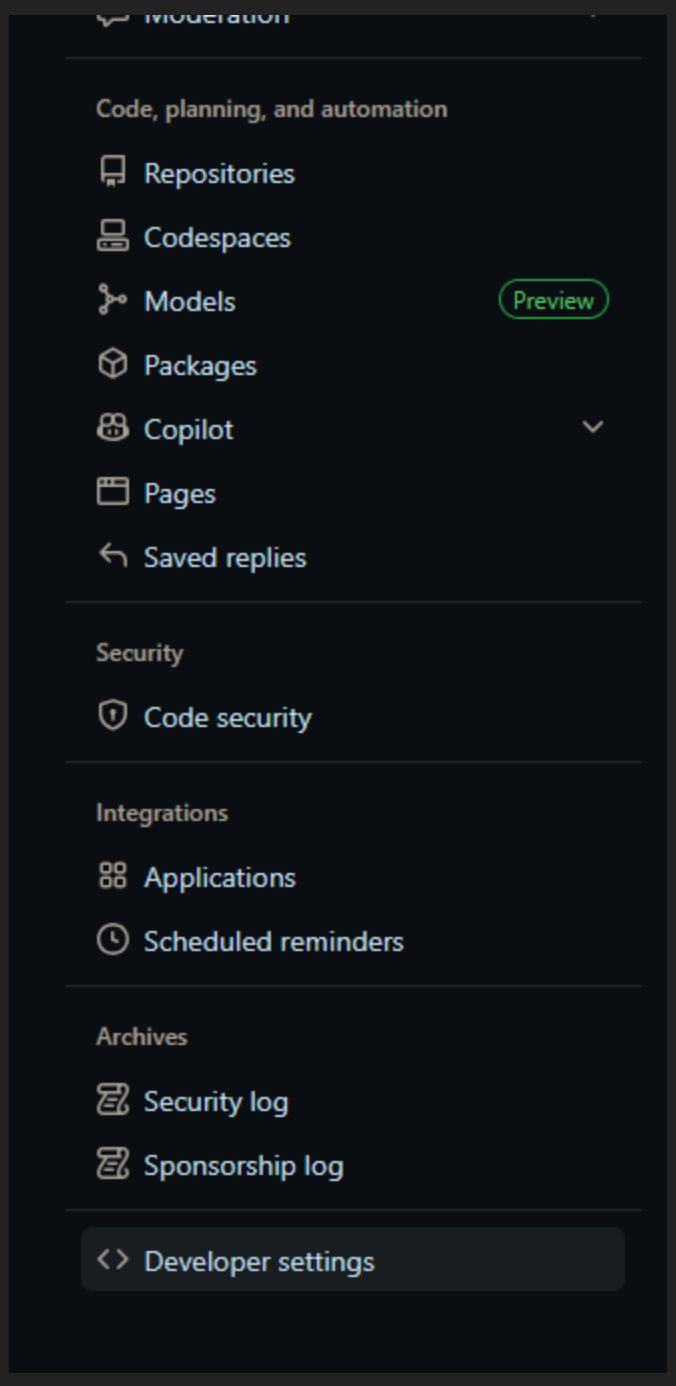
Step 1.

Head to the horde-server package on Epic Games' GitHub, and take note of the docker pull command, along with the latest version available (you can target latest, but it can be helpful to target a specific version until you're ready to update to the next). You will also need to create a Personal Access Token (PAT), to authorize the docker daemon to download the container package.

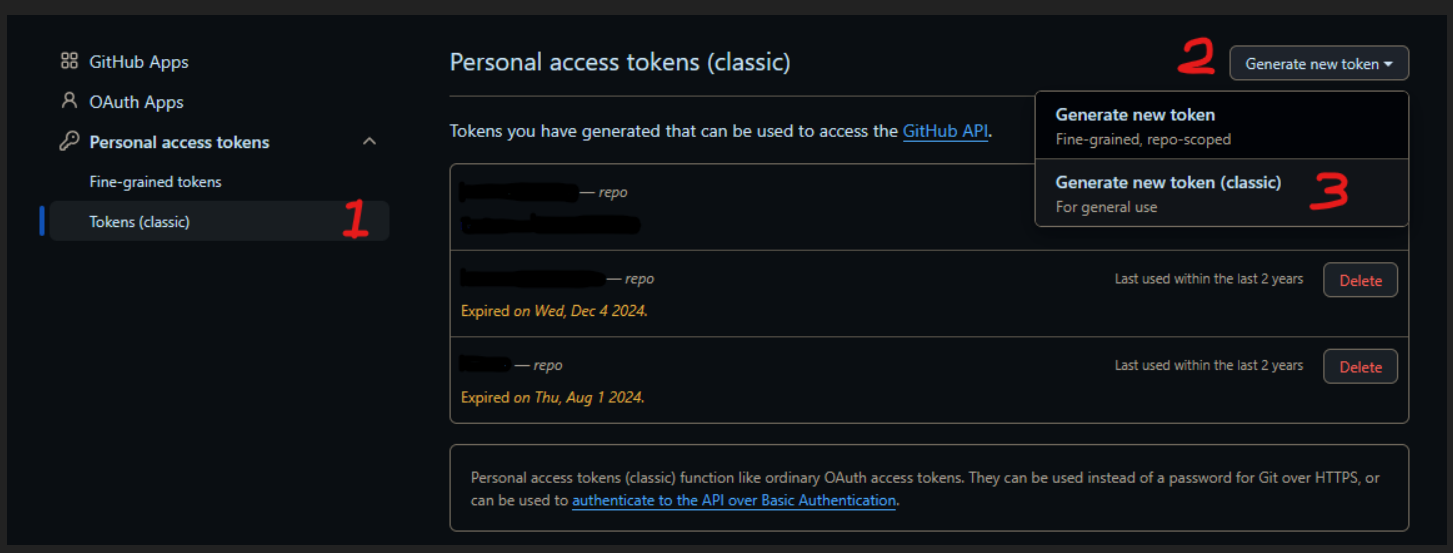
A PAT is created by going to the settings of the account that has access to Unreal Engine's source code repository, Developer settings, and creating a new (classic) token.



Account Settings

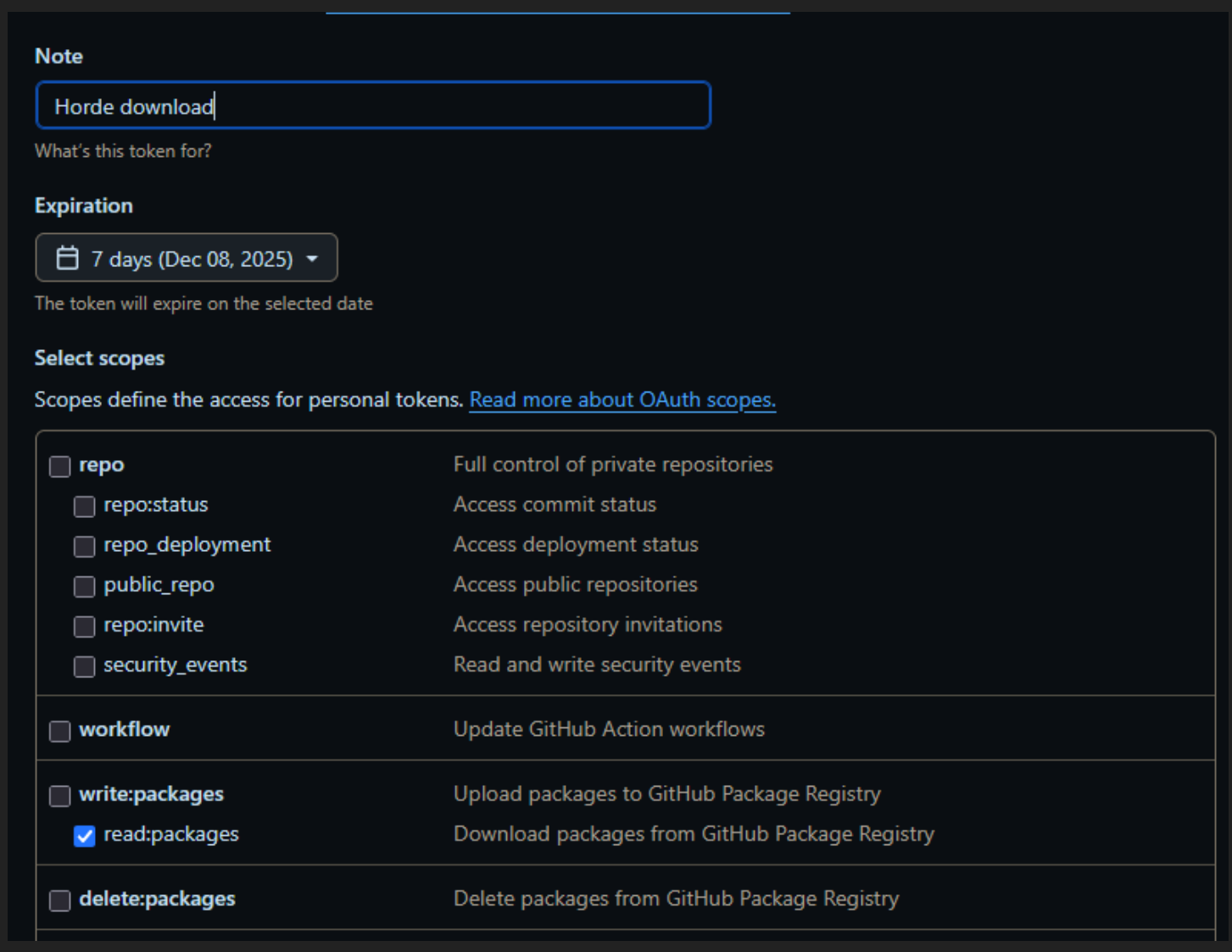


Developer settings

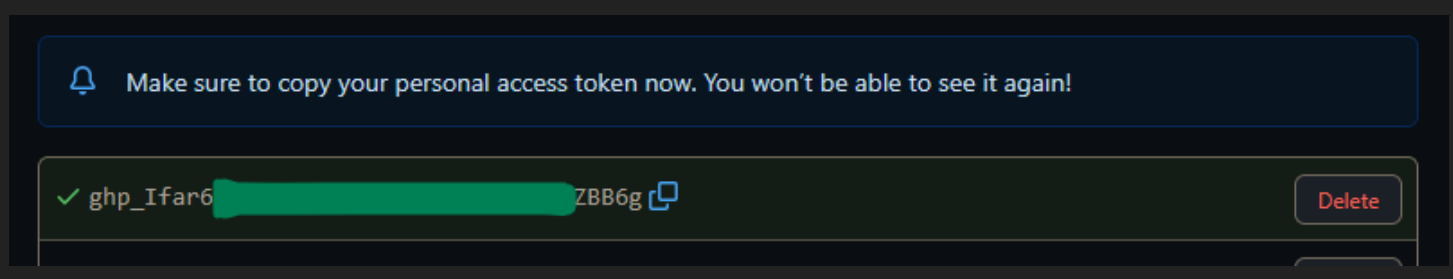


Token Creation

This token only needs to have access to the read:packages permission, and can be set to expire almost immediately after use. As once we've downloaded the package, the token is no longer needed (until you want to download the next version).



This will create your token, make sure to **securely copy and save the token, as it will not be shown again**. Because this token now acts as a password to your account (within the scopes that you've given it).



Step 2.

Now that we have both the container package that we want to download, and the PAT to authorize that download, we can continue with actually downloading the package to the computer that will be running the Horde Server (note that container packages can't easily be copied between computers as a file)

To do this, we run the following commands in a command prompt or terminal:

```
docker login ghcr.io
```

This will prompt you for a login, use your regular email, but for the password, use the PAT that we created in the previous step. This should successfully log you in and authorize access to pull the horde-server package

```
docker pull ghcr.io/epicgames/horde-server:latest
```

Step 3.

With the container package downloaded, we can start defining the `docker-compose.yaml`, which will form the main configuration for the container such as ports to forward, environment variables, and volumes / files to share with the host operating system.

To start with this, we'll first need to create a folder in which we keep all the relevant files, both configuration and eventual data storage for the server to have everything in one place. The folder structure should be like this:

- *.../Path/To/Root*
  - *Data*
  - *Defaults*
  - *Certs*
  - *Storage*
  - `docker-compose.yaml`
  - `appsettings.json`
  - `appsettings.Production.json`

Where `*Data*`, `*Defaults*`, `*Certs*` and `*Storage*` are folders. Not filenames.

this way everything will be able to use paths relative to the Root folder using a period (`.`) at the start of a path, e.g. `./Data` will point to the aforementioned *Data* folder relative to the Root folder in which the `docker-compose.yaml` file exists. The contents of the `docker-compose.yaml` are displayed below:

> Tweak the file according to your requirements (see comments for details)

```yaml
services:
  horde-server:
    # The name of the container package you downloaded in step 2
    image: ghcr.io/epicgames/horde-server:5.7.0
    restart: always
    environment:
      # if changing the password to the mongodb instance, change it here as
      Horde__DatabaseConnectionString: mongodb://horde:dbPass123@mongodb:270
      Horde__HttpPort: 13340
      Horde__Http2Port: 13342
      Horde__RedisConnectionConfig: redis:30002
      # In case you have your own Perforce server that you want Horde to con
      #P4TRUST: /app/Data/p4trust.txt
    ports:
      # the ports to forward from the host into the container, must match th
      - 13340:13340
      - 13342:13342
      #- 13344:13344
    # in case you have local DNS names, they can be added in the extra_hosts
    #extra_hosts:
      #- example.sigma-erebus.com=192.168.0.204
    volumes:
      # Configure the volumes to bind to folders relative to this file for e
      - ./Data:/app/Data
      - ./Defaults:/app/Defaults
      - ./Certs:/home/Horde/Certs
      - ./Storage:/mnt/Storage
      - ./appsettings.json:/app/appsettings.json
      - ./appsettings.Production.json:/app/appsettings.Production.json
  mongodb:
    image: mongo:7.0.5-jammy
    restart: always
    environment:
      # Default username and password, change these!
      MONGO_INITDB_ROOT_USERNAME: horde
      MONGO_INITDB_ROOT_PASSWORD: dbPass123
    command: --quiet --logpath /dev/null
    ports:
      - 27017:27017
    volumes:
      - mongodb:/data/db
  redis:
    image: redis:6.2-alpine
    restart: always
    ports:
      - 30002:30002
    command: redis-server --save 60 1 --port 30002 --loglevel warning
    volumes:
      - redis:/data

# define the volumes used by mongodb and redis. These will not show up as fo
volumes:
  mongodb:
    driver: local
  redis:
    driver: local
```

docker-compose.yaml

Step 4.

Theoretically we have all we need to start the server, but we can't quite start using the server yet. As you might have noticed, in the folder structure in step 3 there were some more files present. And we'll need to configure parts of the server still. But these files will be automatically generated within the Data folder after the first run of the server, after which we can tweak them to our needs before restarting the server.

To start the server and generate the default configurations, open a command prompt or terminal to the Root folder:

```
cd .../Path/To/Root
```

Once there, we can hand the compose file to docker compose, which will pull up (run) the services as defined in the yaml file.

```
docker compose up
```

and to pull down (stop) the services:

```
docker compose down
```

Step 5.

Now that we've ran the server once and generated all the configuration files, we can start tweaking these to our needs. The first big one that will need tweaking is the `server.json` found in the *`Root/Data`* folder. This file configures the functionality of the Horde Server, and will always require the server to reboot before taking effect.

I would also recommend to make sure the computer hosting this has got a statically assigned IP, as this will make connecting to the dashboard from other computers easier. And will prevent breaking an existing connection due to DHCP giving a new IP address.

The `server.json` is shown below. Here again some parts are included that will not be required in a default setup, and therefore commented out, but are included as easy starting points for further configuration (e.g. TLS certificates for HTTPS support)

> Tweak the file according to your requirements (see comments for details)

```
// Changes to this file requires a restart of the server
{
    "Horde": {

        // How to authorize users to log in to the server. Auth is disabled
        // by Horde itself ("Horde") or to use an external OIDC auth provide
        "AuthMethod": "Horde",

        // Url to the server, needs to match the server host protocol with h
        "ServerUrl": "http://horde.example.com",

        // Path to read the global Horde configuration from. This can be a p
        // The specified file is continuously monitored for updates and will
        "ConfigPath": "globals.json",

        // List of Perforce servers for use specifically by the server (eg.
        // Configuration for the server may be stored in a more secure locat
        "Perforce": [
            {
                "id": "TrueNAS",
                "serverAndPort": "ssl:perforce.example.com:1666",
                "credentials": {
                    "userName": "Horde",
                    "password": "password"
                    //"ticket": "ticket (optional)"
                }
            }
        ],
        "plugins": {
//          "build": {
//              "enabled": "true",
//              "perforce": [
//                  {
//                      "id": "TrueNAS",
//                      "serverAndPort": "ssl:perforce.example.com:1666",
//                      "credentials": {
//                          "userName": "Horde",
//                          "password": "password"
//                          //"ticket": "ticket (optional)"
//                      }
//                  }
//              ]
//          },
            "compute": {
                "enabled": "true",
            },
            "storage": {
                "backends": [
                    {
                        "enabled": "true",
                        "id": "HostStorage",
                        "type": "FileSystem",
                        "baseDir": "/mnt/Storage"
                    }
                ]
            },
            "analytics": {
                "enabled": "true",
                "sinks": {
                    "mongo": {
                        "enabled": "true",
```

```
                        "retainDays": "90"
                    }
                }
            }
        },

        // Configure an editor telemetry sink that stores data in the bundle
        // This stores raw telemetry data; aggregated metrics can be compute
        // needing to scan this data, and is expired separately.
        "Telemetry": [
            {
                "Type": "Mongo",
                "RetainDays": 3
            }
        ],

        // Whether to use the local Perforce environment for reading configu
        "UseLocalPerforceEnv": false
    },
    // TLS certificate configuration. AllowInvalid is required if using a se
//  "Kestrel": {
//      "Certificates": {
//          "Default": {
//              "Path": "/home/Horde/Certs/Horde.crt",
//              "KeyPath": "/home/Horde/Certs/Horde.key",
//              "AllowInvalid": "true"
//          }
//      }
//  }
}
```

server.json

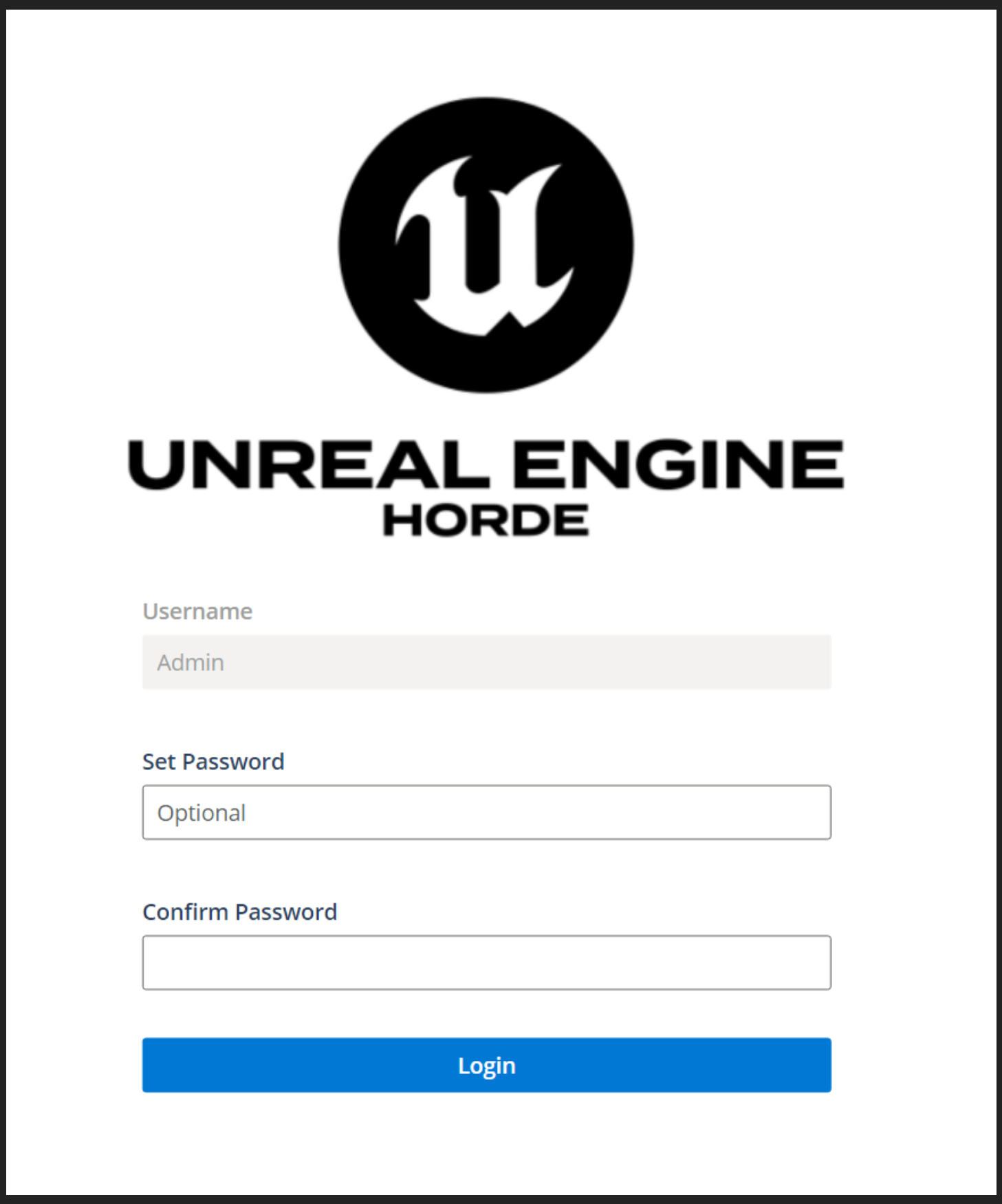
Step 6.

Now that the server.json configuration has been prepared, we can start the server by pulling our compose back up

```
docker compose up
```

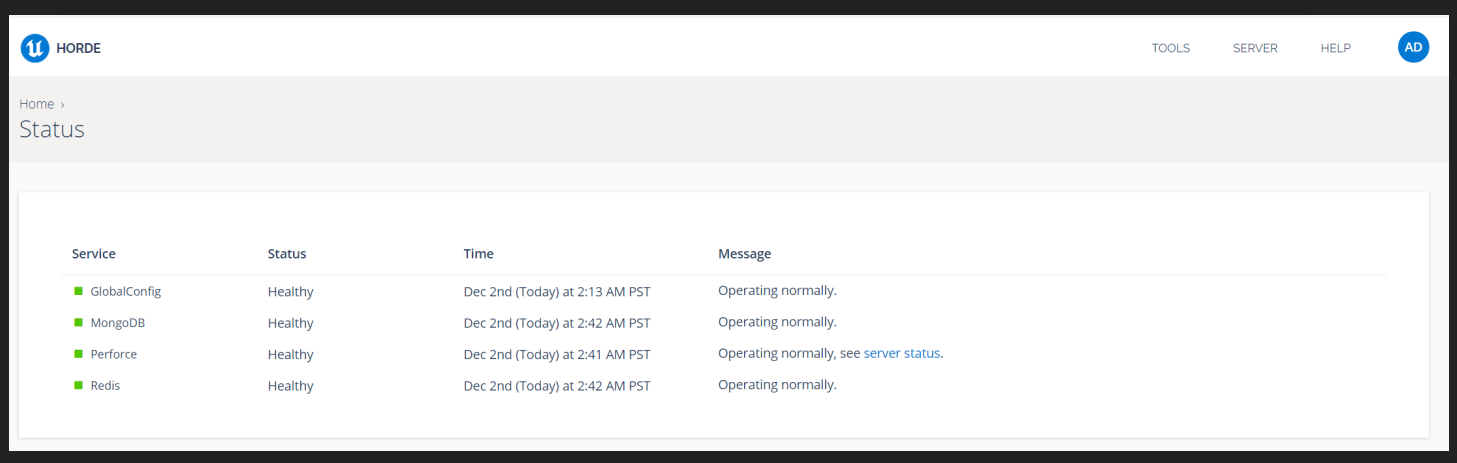
We can now access the online dashboard through a browser by typing the URL to the server:
`http://<your server's IP, or localhost>:<your configured http port>`
This means that usually `http://localhost:13340` will be the required URL to access the dashboard.

Because we've set the authentication method to "Horde" instead of "Anonymous" we will also be prompted to configure your Admin password:



Step 7.

With the `server.json` configuration out of the way, we'll only have to configure the `global.json` file, which can be hot-reloaded by the server, and therefore does not need any reboots. Since it is hot-reloadable, the server also monitors for any configuration problems, and will notify you if it detects any (syntax) errors on its status page, found under `Server > Status`:



The configuration for `global.json` is shown below. In this config, once again a large part has been commented out and left for easy further configuration. The main part of the configuration is around the compute and analytics plugin configurations:

> Tweak the file according to your requirements (see comments for details)

If using Horde authentication, make sure that any non administrator users that are created also have appropriate permissions where relevant. An agent doesn't need `AddComputeTasks` for helping with a task, but for initiating a new task it is needed.

```
{
    // Schema version number for the config file. New Horde versions may bum
    // will be printed to the log on startup.
    "version": 2,

    // Include default settings from the Horde installation folder
    "include": [
        {
            "path": "$(HordeDir)/Defaults/default.global.json"
        }
    ],

    "plugins": {
        "build": {
            // Definitions for projects in the CI system
            "projects": [
//              // Uncomment this to enable a default //UE5/Dev-Main-Horde s
//              {
//                  "id": "ue5",
//                  "path": "ue5.project.json"
//              }
            ],

            // Configuration for different Perforce servers. Horde allows ru
            // selection is performed to evenly distribute load, and servers
//          "perforceClusters":
//          [
//              {
//                  // Name for this cluster. Streams can be configured to u
//                  "name": "Default",
//
//                  // Default account to use inside the Horde Server for qu
//                  // If this value is not specified, the first item from t
//                  "serviceAccount": null,
//
//                  // Whether the service account supports generating logir
//                  // allows Horde to submit changes on behalf of users.
//                  "canImpersonate": false,
//
//                  // List of servers to access
//                  "servers":
//                  [
//                      {
//                          // DNS name and port for the server.
//                          // If connecting using "ssl:" prefix, ensure P4
//                          // See Horde's documentation on connecting to SS
//                          "serverAndPort": "ssl:perforce.example.com:1666"
//                      }
//                  ],
//
//                  // List of credentials for the server. The first credent
//                  "credentials":
//                  [
//                      {
//                          "userName": "Horde",
//
//                          // Specify either the password or ticket propert
//                          "password": "password",
//                          // "ticket": "ticket if not using password"
//                      }
//                  ]
```

```
//              },
//              {
//                  // Name for this cluster. Streams can be configured to u
//                  "name": "Perforce-Server-2",
//
//                  // Default account to use inside the Horde Server for qu
//                  // If this value is not specified, the first item from t
//                  "serviceAccount": null,
//
//                  // Whether the service account supports generating logir
//                  // allows Horde to submit changes on behalf of users.
//                  "canImpersonate": false,
//
//                  // List of servers to access
//                  "servers":
//                  [
//                      {
//                          // DNS name and port for the server.
//                          // If connecting using "ssl:" prefix, ensure P4
//                          // See Horde's documentation on connecting to SS
//                          "serverAndPort": "ssl:perforce2.example.com:1666
//                      }
//                  ],
//
//                  // List of credentials for the server. The first credent
//                  "credentials":
//                  [
//                      {
//                          "userName": "Horde",
//
//                          // Specify either the password or ticket propert
//                          "password": "password",
//                          // "ticket": "ticket if not using password"
//                      }
//                  ]
//              }
//          ]
        },
        "compute": {
            "clusters": [
                {
                    "id": "default",
                    "namespaceid": "horde.compute",
                    "acl": {
                        "entries": [
                            {
                                "claim": { "type": "http://epicgames.com/ue/
                                "actions": ["AddComputeTasks"]
                            }
                        ]
                    }
                }
            ]
        },
        "analytics": {
            "stores": [
                {
                    "id": "engine",
                    "include": [
                        {
                            "path": "$(HordeDir)/Defaults/default-metrics.te
                        },
                        {
                            "path": "$(HordeDir)/Defaults/default-views.tele
```

```
                    }
                ]
            }
        ]
    }
},

//  // Default deployment parameters for other tools bundled with Horde. The
//  "parameters": {
//      "ugs":
//      {
//          // Set the default Perforce server for anyone using UGS.
//          "defaultPerforceServer": "ssl:perforce.example.com:1666"
//      }
//  }
}
```

global.json

Step 8.

After making these tweaks, there is another tweak that I would recommend to the `default.global.json` that is loaded at the top of the `globals.json` file. this file can be found within the *`Defaults`* folder as defined in the compose file. The part we'll be tweaking is the pools definitions. By default the pools are configured like this:

```
"pools": [
    {
        "name": "Win-UE5",
        "condition": "Platform == 'Win64'",
        "color": "Blue"
    },
    {
        "name": "Win-UE5-GPU",
        "condition": "Platform == 'Win64' && Interactive == true",
        "color": "Blue"
    },
    {
        "name": "Mac-UE5",
        "condition": "Platform == 'Mac'",
        "color": "Green"
    },
    {
        "name": "Linux-UE5",
        "condition": "Platform == 'Linux'",
        "color": "Orange"
    }
]
```

pool definitions within "plugins" > "compute"

These pools will be assigned automatically based on the capabilities of the agent that connects. These pools are also the things targeted when initiating a compile. So when starting a compile to the `Win-UE5` pool, Linux machines that are capable of running Windows compiles through Wine will not be utilized. To fix this, I recommend rewriting the pool definitions to this, I have attached the full `default.global.json` below for easy copy-paste:

```
"pools": [
    {
        "name": "Win-UE5",
        "condition": "Platform == 'Win64'",
        "color": "Blue"
    },
    {
        "name": "Win-UE5-GPU",
        "condition": "Platform == 'Win64' && Interactive == true",
        "color": "Blue"
    },
    {
        "name": "Win-UE5-Compatible",
        "condition": "Platform == 'Win64' || WineEnabled == true",
        "color": "Blue"
    },
    {
        "name": "Mac-UE5",
        "condition": "Platform == 'Mac'",
        "color": "Green"
    },
    {
        "name": "Linux-UE5",
        "condition": "Platform == 'Linux'",
        "color": "Orange"
    },
    {
        "name": "Linux-UE5-Wine",
        "condition": "Platform == 'Linux' && WineEnabled == true",
        "color": "Orange"
    }
]
```

updated pool definitions within "plugins" > "compute"

```
{
    // Settings for the dashboard
    "dashboard": {

        // Whether to show the default landing page with documentation when
        "showLandingPage": true,

        "showCI": true,
        "showAgents": true,
        "showPerforceServers": true,
        "showDeviceManager": true,
        "showTests": true
    },

    // Settings for various plugins
    "plugins": {

        "compute": {
            "clusters": [
                //{
                //  "id": "default",
                //  "namespaceId": "horde.compute"
                //}
            ],
            "networks": [
                {
                    "id": "default",
                    "cidrBlock": "0.0.0.0/0",
                    "description": "Default network (matching all IPs)",
                    "computeId": "default"
                }
            ],
            "pools": [
                {
                    "name": "Win-UE5",
                    "condition": "Platform == 'Win64'",
                    "color": "Blue"
                },
                {
                    "name": "Win-UE5-GPU",
                    "condition": "Platform == 'Win64' && Interactive == true
                    "color": "Blue"
                },
                {
                    "name": "Win-UE5-Compatible",
                    "condition": "Platform == 'Win64' || WineEnabled == true
                    "color": "Blue"
                },
                {
                    "name": "Mac-UE5",
                    "condition": "Platform == 'Mac'",
                    "color": "Orange"
                },
                {
                    "name": "Linux-UE5",
                    "condition": "Platform == 'Linux'",
                    "color": "Green"
                },
                {
                    "name": "Linux-UE5-Wine",
                    "condition": "Platform == 'Linux' && WineEnabled == true
                    "color": "Green"
```

```
            },
        ]
    },

    "storage": {
        "backends": [
            {
                "id": "default-backend",
                "type": "FileSystem",
                "baseDir": "Storage" // Default base directory is C:\Pro
            },
            {
                "id": "memory-backend", // Used for automated tests
                "type": "Memory"
            }
        ],
        "namespaces": [
            {
                "id": "default",
                "backend": "default-backend"
            },
            {
                "id": "horde-artifacts",
                "prefix": "Artifacts/",
                "backend": "default-backend"
            },
            {
                "id": "horde-perforce",
                "prefix": "Perforce/",
                "backend": "default-backend"
            },
            // The 'horde-logs' namespace is used internally by Horde to
            // and remote execution systems. It must be configured in or
            // correctly.
            {
                "id": "horde-logs",
                "prefix": "Logs/",
                "backend": "default-backend"
            },
            {
                "id": "horde-tools",
                "prefix": "Tools/",
                "backend": "default-backend"
            },
            {
                "id": "memory",
                "backend": "memory-backend",
                "enableAliases": true
            }
        ]
    },

    "build": {
        "artifactTypes": [
            {
                "name": "step-output",
                "keepDays": 1
            }
        ],
        "devices": {
            "platforms": [
                {
                    "id": "android",
                    "name": "Android"
```

```
                }
            ],
            "pools": [
                {
                    "id": "ue5",
                    "name": "UE5",
                    "poolType": "Automation"
                },
                {
                    "id": "remote-ue5",
                    "name": "Remote UE5",
                    "poolType": "Shared"
                }
            ]
        }
    }
},

// Default ACL groups
"acl": {
    "entries": [
        {
            "claim": {
                "type": "http://epicgames.com/ue/horde/group",
                "value": "View"
            },
            "profiles": [
                "default-read"
            ]
        },
        {
            "claim": {
                "type": "http://epicgames.com/ue/horde/group",
                "value": "Run"
            },
            "profiles": [
                "default-run"
            ]
        }
    ]
}
}
```

full default.global.json

These conditions can also be customized to evaluate custom properties in an agent's configuration to pool agents in more nuanced ways. These custom properties can be defined in the `agent.json` file (details will be in agent setup guides). But once configured, the conditions can also evaluate based on those custom keys to have more compute pools:

```
{
        "name": "Team-A",
        "condition": "Team == 'Team-A'",
        "color": "Grey"
},
```

if desired, colors here can be tweaked as well (my personal preference is to swap around the colors of Mac & Linux). However for colors you can only pick from: Blue, Orange, Green, Grey and Default.

Member discussion

0 comments

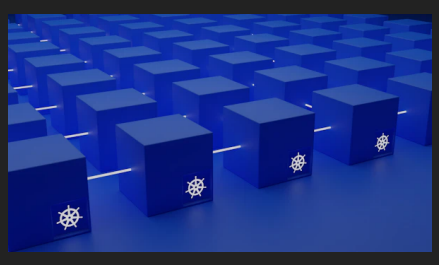
Start the conversation

Become a member of **Development Blog Hidde Derks** to start commenting.

Sign up now

Already a member? Sign in

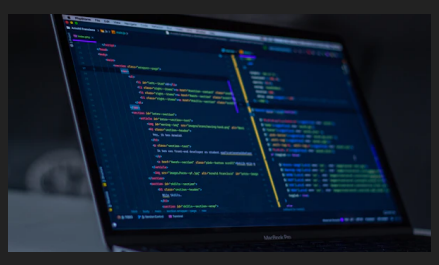
READ MORE



Unreal Horde Agent Deployment (Docker)

Initially created for the Breda University of Applied Scienc...

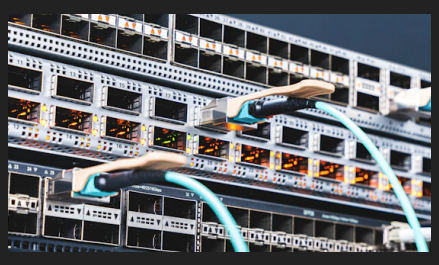
By Hidde Derks — 02 Dec 2025



Unreal Horde Agent Deployment (Windows)

Initially created for the Breda University of Applied Scienc...

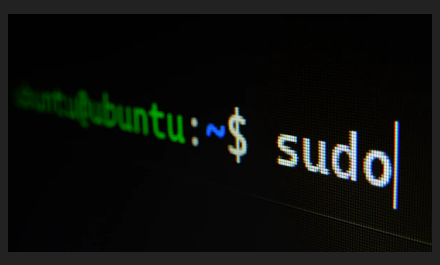
By Hidde Derks — 02 Dec 2025



Unreal Horde Server Deployment (Windows)

Initially created for the Breda University of Applied Scienc...

By Hidde Derks — 02 Dec 2025



Use Powershell to run a command as another user

In Jenkins, every now and then you might need to work...

By Hidde Derks — 25 Jun 2025

Development Blog Hidde Derks

Development blog of Hidde Derks (a.k.a. Sigma-Erebus)

jamie@example.com Subscribe