DOC: appliance/webaccelerator/backends/index.txt

Backend
Backend Monitoring
The Web Accelerator constantly monitors the operation of back-ends, checking that requests for a specific URL return a 200 status.
When monitoring reports an error, the Web Accelerator stops sending requests to that back-end. If there are no other functioning back-ends, it will continue to respond with the objects remaining in cache for a certain period after their expiration, and then with a courtesy page. In this way short outages or maintenance of a cacheable site can remain invisible to visitors.

Normally Seeweb staff install a `ping.php` file
on the site,
or on a dedicated domain if the back-end serves multiple sites.
For sites managed with WordPress, the file automatically also monitors the database server's operation. In any case the customer can provide a monitoring URL that fits their needs.

Courtesy Page
If all back-ends are unreachable and there is no usable response in the cache, the Web Accelerator replies with a courtesy page.
If desired, it can be customized with your own graphics.
Because the courtesy page must work even when the back-end is unavailable, our staff will inline any CSS and images. We recommend keeping it around 10 kB in size.

Back-end Load Balancing
Redundancy and scalability of back-ends can be handled with a load-balancing mechanism: the Web Accelerator can forward traffic to multiple back-ends and ignore those that are not functioning.

Multiple Back-ends
For complex sites where different parts of the same domain are served by different back-ends, the Web Accelerator can route requests to the appropriate back-end based on the URL or other characteristics.

Handling Visitors' Real IPs
The back-end sees all requests coming from the Web Accelerator's IP and can use the standard header `X-Forwarded-For` to obtain the visitor's real IP.
If the back-end uses Apache, our staff will configure `mod_rpaf` or `mod_remoteip` to handle it transparently.
For back-ends with IIS you can set:

```
Add-WebConfigurationProperty -pspath 'MACHINE/WEBROOT/APPHOST' ↩
-filter "system.applicationHost/sites/siteDefaults/logFile/↩
customFields" -name "." -value @{logFieldName='X-Forwarded-For';↩
sourceName='X-Forwarded-For';sourceType='RequestHeader'}
```

```

### HTTPS Request Handling
The Web Accelerator also acts as a TLS terminator, so requests originally made with HTTPS reach the back-end always via HTTP and can be identified by the header `X-Forwarded-Proto: https`.

In the default configuration the Web Accelerator itself performs the redirect from HTTP to HTTPS, so the back-end can be configured to use only HTTP and does not need to handle redirect logic directly.

If the CMS needs to distinguish HTTP from HTTPS requests, you can integrate it with this Apache directive, which our staff usually already installs in the server's global configuration:

```
SetEnvIf X-Forwarded-Proto "^https$" HTTPS=on
```

If you use nginx instead, you can configure, for example, in `/etc/nginx/fastcgi.conf`:

```
fastcgi_param X_FORWARDED_PROTO $http_x_forwarded_proto if_not_empty;
```

Rarely, some WordPress plugins require inserting code like the following directly into `wp-config.php`:

```php
/* For HTTPS handling via the Web Accelerator -- Seeweb */
if (isset($_SERVER['HTTP_X_FORWARDED_PROTO'])
    && $_SERVER['HTTP_X_FORWARDED_PROTO'] == 'https')
    $_SERVER['HTTPS'] = 'on';
```

Timeout
The Web Accelerator implements timeouts at several levels to terminate requests that take too long. We recommend not increasing them globally in order to avoid making the service more vulnerable to denial-of-service attacks.

Our staff is available to investigate excessively short timeouts, but the most common case is the timeout that fires when the first byte of the server's response is not received. It can be increased, but this often indicates that the application is poorly written and buffers a large response instead of streaming it as it is generated.

==

DOC: appliance/webaccelerator/caching/index.txt

```

Caching
Caching Modes
A caching policy must be decided: the goal is to cache as many responses as possible (to increase efficiency), but never cache when it is not allowed (to avoid breaking site functionality).

In the default mode the Web Accelerator follows the normal HTTP semantics. This mode is usually not useful, at least without custom configuration, because HTTP forbids caching of responses that receive or set cookies, and virtually all requests involve cookies.

Standard caching: as an evolution of the previous mode, cookies can be ignored for objects presumed to be static, thereby forcing their caching: images, CSS, JavaScript, fonts, etc. This configuration works without any customizations, but it does not cache dynamic pages and therefore does not deliver the best performance.

Forced caching: all objects are cached except those explicitly excluded, such as the site's administrative area or the shopping cart. This requires detailed knowledge of the CMS or the specific site, but a careful configuration yields an optimal caching level. The Web Accelerator automatically handles the most common cases thanks to a library of ready-made configuration modules for the most popular CMSs such as WordPress, Joomla, and Magento.
It is also possible to use criteria more complex than the URL to exclude something from caching: for example, the Web Accelerator can examine cookies to know if a request was made by an authenticated user, who therefore must receive a personalized page.
Furthermore, the information contained in cookies can be used to partition the cache and have multiple copies of the same page personalized according to characteristics such as language, currency, or even each authenticated user.
It is understood that normally caching is performed only for responses to requests with the HTTP **GET** or **HEAD** methods.

TTL of cached objects
The Web Accelerator caches objects for the time specified by the **Cache-Control** headers returned by the backend, up to a maximum duration (default: 1 day).
If the header is missing, a default TTL is applied based on the file extension in the URL: for presumed static objects (images, CSS, JavaScript, fonts, etc.) this value is **1 hour**, while for presumed dynamic pages (i.e., everything else) it is **5 minutes**. Redirects **302** and **307** and errors **403** and **404** are instead cached for **1 minute**.
If the backend cannot send appropriate **Cache-Control** headers, it is still possible to configure custom policies on the Web Accelerator with maximum granularity. Our staff is available to assess the best settings for each site.
If the backend is down and therefore the cache cannot be refreshed, the Web Accelerator serves expired objects up to **one hour** old.
It should be noted that if the backend sends objects with a header such as **Cache-Control: no-cache**, caching will be prevented: in this case,

if it cannot be fixed, the Web Accelerator can be configured to ignore them completely.

Cache invalidation
Three different APIs are available to invalidate cache content:

- Requests for specific objects using the HTTP **PURGE** method.
- Objects matching a regular expression requested with the HTTP **PURGE** method, adding a header `X-Purge-Method: regex`.
- The Drupal invalidation API, without needing to set `X-Varnish-Secret`.

Requests are accepted only from the backend IP address, or optionally from other configured authorized addresses. **PURGE** requests can be sent with either HTTP or HTTPS, whereas **BAN** requests must be sent with HTTPS.

To automatically handle invalidation in WordPress, you can install a plugin that implements one of these interfaces, for example **Proxy Cache Purge**.

A sample file `test_purge.php` is available that shows how to invalidate an object with PHP and libcurl.

Temporary disabling of caching
During development activities that directly involve the production site, caching can be temporarily disabled for the current browser session by accessing

```
https://<domain>/seeweb-wa/cache-bypass
```

Informational headers
Varnish responses contain these headers that allow you to understand where the object comes from:

```
x-varnish: 5217771 2207419
age: 261
```

If **X-Varnish** contains two numbers it means the response comes from the cache, whereas if it contains only one it means it comes from the backend.
The **Age** header indicates the age in seconds of the returned object.

==

DOC: appliance/webaccelerator/certificates/index.txt

TLS Certificates - Use of TLS

In the default configuration the Web Accelerator redirects to HTTPS for all requests received via HTTP. This is mandatory when the WAF function is enabled. Commercial certificates purchased by the client can be installed manually, or they can be issued and renewed automatically with **Let’s Encrypt** or other Certificate Authorities that use the ACME protocol.

Certificate management with ACME
Normally the Web Accelerator uses the **HTTP-01** authorization mechanism to obtain certificates from Let’s Encrypt. This requires that the domain already points to the Web Accelerator at the time of the request, so to avoid downtime our staff can temporarily install a certificate fetched from the backend, which will be automatically replaced in the following hours by a new one issued by Let’s Encrypt.

If it is not possible to obtain a working certificate for a domain that is being transferred to the Web Accelerator, and you do not wish to purchase a commercial one, you can coordinate with our staff to request a Let’s Encrypt certificate immediately after the DNS has been updated, to minimize disruption for visitors.

If a wildcard certificate with Let’s Encrypt is required, you can use the **DNS-01** authorization mechanism by creating in your DNS zone a record such as

```
_acme-challenge.example.com : _acme-challenge.example.com. ↩
CNAME example.com.acme-wa.seeweb.it.
```

==========

DOC: appliance/webaccelerator/introduction/index.txt

Introduction
Caching
Caching is essential for the scalability of a dynamic website. Depending on the complexity of the site, it becomes increasingly difficult to provide good performance for requests that end up being handled by PHP. In general, there are orders-of-magnitude differences between the latency of a response that comes from the cache and one that comes from the backend. Moreover, a caching mechanism may be necessary to mitigate denial-of-service attacks.
The Web Accelerator combines configuration modules for Varnish that have been developed and refined over years of experience with the customizations needed for the client’s site to achieve the highest possible efficiency. It is a product fully managed by Seeweb staff, without the client needing to configure it personally.

Web Application Firewall

The Web Accelerator platform can act as a frontend cache, a Web Application Firewall (WAF), or both simultaneously.
The WAF uses a set of proprietary rules created by Seeweb staff to block the most common intrusion attempts.
Adopting a WAF is often a regulatory requirement, but our staff can also implement custom configurations to protect specific sites with known security issues that cannot be updated.
All the functions described here apply to both products.

Manageable Sites
A Web Accelerator can manage an arbitrary number of sites, provided they all use the same CMS. Ready-made configurations exist for the most common CMSs; otherwise Seeweb staff will develop a custom configuration at no extra cost.
To activate new sites, simply contact our staff so they can provision the necessary TLS certificates.

Web Accelerator Technology
- Caching: Varnish.
- TLS termination: nginx.
- WAF: ModSecurity with proprietary rules.
- Configuration engine: YAML, Perl, Template::Toolkit, Moose, Moo.
- Observability: Munin, Prometheus.
- Real-time statistics: goaccess.
- Operating system: Debian.
- Infrastructure: Seeweb Cloud Servers.

==

DOC: appliance/webaccelerator/more/index.txt

Additional Functions
High Availability
- Local: two instances can be activated in the same data center in active/passive mode.
- Geographic anycast: instances can be activated in different data centers, each handling a portion of the traffic, and in case of failure or maintenance one can serve as the backup for the other.
Seeweb has data centers in Milan and Frosinone with independent connectivity and peering, allowing low-latency service for local visitors.
For ultra-high-traffic sites it is also possible to activate multiple instances in the same data center in active/active mode, but this is rarely needed.

Real-time Statistics
https://<domain>/seeweb-wa/statistics
shows real-time traffic statistics for the entire Web Accelerator.
Credentials are provided at activation.

Emergency Caching

Sometimes a client’s site suddenly receives an unexpected amount of traffic, legitimate or part of a denial-of-service attack.
Seeweb staff can activate an emergency Web Accelerator within minutes, diverting HTTP and HTTPS traffic with flowspec that would normally go to the backend. This avoids having to change the site’s IP in DNS and wait for all clients to see the new address.
The emergency cache is a near-standard Web Accelerator. It can forward traffic directly to the backend for sites it is not configured to cache, so only the certificates for the sites that actually need caching must be installed.

==

DOC: appliance/webaccelerator/options/index.txt

Options
These are some of the configuration modules available for the Web Accelerator.

Mobile/Desktop Browser Distinction
The Web Accelerator can be configured to cache pages requested by desktop browsers separately from those requested by mobile browsers. This allows caching even for legacy sites that perform server-side browser sniffing. If the backend emits a `Vary` header that prevents caching, the Web Accelerator can rewrite it.

Rate Limiting
The module allows defining thresholds beyond which requests are rejected with a temporary 429 error. For example, a limit can be applied to the number of requests from the same IP to the same domain.
Normally limits are applied only to requests that would reach the backend, but they can also be applied to requests that would be served from the cache.

Blocking Undesired Clients
The module examines the `User-Agent` header to block certain types of scans, aggressive crawlers, or clients that bring no value to the site. It is reasonably safe to use because the same configuration protects over 100,000 Seeweb client sites.

Botguard
Web Accelerator and WAF can be integrated with the Botguard service for protection against scraping and other non-human clients.

Cookie Filtering
It is possible to manipulate cookies sent to the backend to remove those that would prevent caching.

Authorization with Cookie
The module allows excluding from caching requests that contain a specific authentication cookie, optionally only for certain URLs.

Edge Side Includes (ESI)
The Web Accelerator can use a simple markup language to compose a page from multiple backend requests. In this way a single page can be built from cacheable elements and dynamic ones.

Geoblocking or Geo-allowing
Requests from certain countries can be blocked, or only requests from listed countries can be accepted.
If this function is enabled restrictively, it is important to ensure that search-engine crawlers are not blocked.

Prevent Hotlinking
This module prevents image hotlinking.
Usually it is advisable to configure it to still accept requests with a `Referer` header from the most common search engines.

WebP Negotiation
Some sites implement handling of images in WebP format through server-side negotiation, to adapt to the capabilities of the visitor's browser.
A site that changes its response based on secondary request characteristics would not be compatible with a frontend cache, but this module solves the problem by implementing WebP negotiation directly in the Web Accelerator.

Caching of OPTIONS responses
If the site structure allows, it is possible to enable caching of responses to OPTIONS requests generated by the Cross-Origin Resource Sharing (CORS) mechanism.

fail2ban
fail2ban can be enabled on the Web Accelerator to block repetitive attacks before they reach the backend. It is important that it is not active on the backend, since all traffic comes from the Web Accelerator's IP and therefore it could not block it.

Log Management
By design, the Web Accelerator does not retain statistics or request logs. To monitor the traffic handled, you can view the network graphs in the cloud server panel associated with the Web Accelerator. If you want a web analytics system, we recommend installing Matomo or Plausible. For real-time analysis, we can send the request stream via syslog to a SIEM or other client log analyzer. Alternatively, the Web Accelerator can upload each night the day's request log to an S3-compatible service such as Seeweb's cloud object storage.

==

DOC: en/index.txt

Overview

Seeweb is the Italian Cloud Computing Provider, part of the Dominion Hosting Holding group.
The first company to introduce cloud hosting services in Italy in 2009, ushering in the cloud era, Seeweb has also pioneered Artificial Intelligence and Machine Learning in Italy, with its cloud-based GPU and NPU Computing infrastructures.
With its multiple proprietary Data Centers and support from DHH points of presence, Seeweb facilitates the design and creation of scalable, flexible, redundant IT architectures, with a minimum uptime of 99.9% with penalties guaranteed by SLA.
Its Housing and Colocation services enable businesses to place their servers and equipment on efficient infrastructures dedicated to a maximum balance between performance and sustainability, thanks to specific energy consumption optimization, recognized by ISO14001 certification and acknowledgment as a green provider by The Green Web Foundation.
A wide range of cloud services guarantees service continuity and speed for management systems, databases, applications, and software of any kind: our cloud servers allow for horizontal and vertical scalability and a modularity that flexibly follows the developments of your projects, including those for AI and ML: you will find in our GPU Cloud and NPU Server services a technology capable of combining top performance with maximum flexibility, with rapid and on-demand provisioning of computing resources.
Together with technology from carefully selected top vendors and the highest quality and sustainability of hardware components, Seeweb services are built to serve complex and multicloud projects, supported by a Presales Engineering team and a Service Desk focused on ensuring proactive and effective technical assistance.
Our IaaS and PaaS solutions are designed to offer a simple user experience, free from any form of technological lock-in and integrable with any infrastructure thanks to technologies like Kubernetes.
In the following guides, you will learn how to use our Cloud products and services to facilitate your daily use of them.

==

DOC: hosting/cloudserver/Cloud-Server/get-started/index.txt

Main Activities
Cloud Servers are virtual Linux or Windows instances that correspond to fully autonomous operating systems.

Activating a New Server
From the Cloud Server section:

- Specify in which Data Center to activate the new instance among:
  - Frosinone, Italy;
  - Milan, Italy;
  - Lugano, Switzerland.
- Choose the desired operating system.
- Select the desired resources (Disk, RAM, and CPU).

Info
SPU (Seeweb Power Units) is the measurement unit used to identify the daily consumption of Cloud services resources (RAM, disk, processors); the minimum increment of additional SPU beyond those included in the product’s base configuration is 100 units, counted on a daily basis.

Next, select the desired Security services.
Select the type of support and optional services.

Connecting to a Server
Access to a server occurs in two modes:

- **SSH** for Linux.
- **RDP** for Windows Server.

Connecting to a Linux Server
To connect to your Linux instance, use an SSH client via a terminal:

- open the terminal and type the command:

  ```bash
  ssh root@xxx.xxx.xxx.xxx
  ```

  Replace `xxx.xxx.xxx.xxx` with the IP address of your instance; confirm with `yes`; if you are not using an SSH key, you will be prompted for the login password.

Note
The password is sent by email and must be changed upon first login to the server.

Info
Windows users can use PuTTY.

Connecting to a Windows Server
To connect to a Windows server you need an RDP client, usually included in Windows 7, 8, 10, and 11.

- Open the RDP client.
- Enter the IP address of your instance.
- Accept the connection.
- Enter the login credentials.

==

DOC: hosting/cloudserver/Cloud-Server/index.txt

Overview
Cloud Server is a service based on virtual machines that can offer high-level performance, optimized for cloud architecture. Thanks to this

service, we are able to activate virtual server instances that have the following characteristics:
- independent operation with its own stand-alone operating system;
- vertical scalability;
- full virtualization;
- availability in multiple Seeweb regions;

Instance characteristics
The main advantages of this type of cloud server service can be summed up in the formula: reliability, scalability, accessibility.

Notes
The Cloud Server service offers guaranteed dedicated resources.
The instances provided with Cloud Server technology include:
- protection from DDoS attacks;
- DNS management;
- cloud firewall management;
- performance monitoring tools and metrics.

The networks are protected from external cyber attacks by the latest state-of-the-art solutions, guaranteed by Seeweb’s high quality standards, in order to optimize costs and maximize product performance.

Cloud Server use cases
Cloud Server therefore easily adapts to the needs of web agencies, system integrators, developers, e-commerce managers, end-users, and anyone who needs to manage a web project for one or multiple sites or applications.

Production environments:
- Any type of e-commerce software: Magento, PrestaShop

Staging and development environments:
- Applications of any kind.
- Workloads of any size.

DOC: hosting/cloudserver/Cloud-Server/pannello/index.txt
Management panel
With Seeweb Cloud Server, any type of IT project has no growth limits nor rigid computational settings thanks to both vertical scalability — which allows adding CPU, RAM and storage resources — and horizontal scalability — which enables creating complex architectures by building multiple cloud server infrastructures.

Login to management panel
When a Cloud Server product is activated, the access credentials for the machine and, simultaneously, the credentials to access the management panel are sent by email.
By visiting https://cloudcenter.seeweb.it/ and entering the provided credentials, you access the product management panel.

- **Dashboard**
- **Console**

- **Info**

Clicking the “Console” button will give you access to the machine via a web console.

Warning
Clicking the red button on the right shuts down the machine.

Clicking “Configure” opens a window that allows you to modify the machine’s resources:

Provisioning
Core provisioning
Note: The number entered will be the total number of cores for the machine; therefore, if the machine has 2 cores and you want to add 4, you must write the final core count, i.e., 6.

RAM provisioning
Note: The number entered will be the total RAM for the machine; therefore, if the machine has 2 GB of RAM and you want to add 4, you must write the final GB amount, i.e., 6.

Disk space provisioning
Note: Disk space can be added in blocks of 10 GB.
Note: The number entered will be the total disk space for the machine; therefore, if the machine has 20 GB and you want to add 40, you must write the final GB amount, i.e., 60.

Provisioning execution
After modifying the resources, you must choose whether to perform the provisioning immediately (with a corresponding reboot) or schedule it for a specific date/time.

Note: RAM and CPU provisioning require a machine reboot to be applied, are automatic, and can be scheduled at any time.
Note: Disk provisioning must be performed manually by a technician; therefore it is recommended to schedule it between 07:00 and 23:00.
Note: If you urgently need to add resources to the server and increase disk space, it is advisable to request two separate provisionings. The first provisioning request should be for the resources, so it can be fulfilled autonomously, followed by the disk-space request. It is important to follow this order because only one provisioning request can be submitted at a time, and a new request can be made only after the previous one has completed.

Adding services
Clicking “Update” allows you to change the support level on the machine:
- Unmanaged
- Basic
- Global
- Proactive

Or you can add a Backup type.

DOC: hosting/cloudserver/Cloud-Server/ssh-key/index.txt
Configuring an SSH Key
In this section you can manage SSH (Secure Shell) keys to access the service via terminal. The servers that have been created are accessible through a secure, encrypted terminal connection. To enable this type of access, you need to generate and configure a digitally signed certificate on your own computer. Detailed instructions are as follows: the first time you configure this service, you will need to create a new SSH key by clicking the appropriate button.

Creating a new SSH Key
Access to Linux instances can be done via password or SSH key (Secure Shell), which represents the most secure method to access your server.

To use it you must:
- Open the terminal on your computer.
- Run the command

  ```bash
  ssh-keygen
  ```

  and press **Enter**.
- Choose a name for your key (e.g., `id_rsa_seeweb`) and press **Enter**.
- Select a passphrase for access or leave it empty.
- Wait a few moments for the process to complete.

Note
If this procedure is performed multiple times, the key file will overwrite the one generated previously.
At the end, two files will be available:

- `id_rsa_seeweb`
- `id_rsa_seeweb.pub`

Uploading the Public Key
After generating the key, copy the contents of the `id_rsa_seeweb.pub` file into the designated text area labeled *Add your key content here...*, assign a name to the key, and click **Add key** to confirm the insertion.

Deleting the Public Key
Unused SSH keys can be removed from the panel.
- From the context menu: click **Delete**.
- Confirm the operation.

==
========

DOC: hosting/cloudserver/cs-gpu/introduction/index.txt

Overview
GPU Cloud Servers, like Shared-CPU Cloud Servers, are managed through the Ermes web panel and the ECS API service. This means that all functions available for Shared-CPU Cloud Servers via the panel are also available for GPU instances.

Viewing GPU Instances
To view GPU instances, go to:
`Compute → GPU servers`
or navigate to: **Cloud Server GPU**.
The page displayed will be similar to the view for Shared-CPU Cloud Servers.

Creating a New GPU Instance
To create a new GPU Server, click **NEW SERVER** on the GPU Servers overview page or go directly to: **GPU Configurator**.
The Configurator will open.
The configuration process is identical to that for creating Shared-CPU Cloud Servers, with the only difference being the plan selection:
In addition to choosing a plan, you can also configure the number of GPUs required. To do this, click on the GPU count and select the desired value from the available options.
To continue the configuration, you can follow: **Configure a Shared CPU**.

==========

DOC: hosting/cloudserver/cs-highmemory/introduction/index.txt
---

Overview
High-Memory Cloud Servers, like Shared-CPU Cloud Servers, are managed through the Ermes web panel and the ECS API service. This means that all functions available for Shared-CPU Cloud Servers via the panel are also available for High-Memory instances.

Viewing High-Memory Instances
To view High-Memory instances, go to:
`Compute → High Memory servers`
or navigate to: **Cloud Server High Memory**.
The page displayed will be similar to the view for Shared-CPU Cloud Servers.

Creating a New High-Memory Instance
To create a new High-Memory Server, click **NEW SERVER** on the High-Memory Servers overview page or go directly to: **High Memory Configurator**.

The Configurator will open.
The configuration process is identical to that for creating Shared-CPU Cloud Servers, with the only difference being the plan selection: only High-Memory plans will appear.
To continue the configuration, you can follow: **Configure a Shared CPU**.

==

DOC: hosting/cloudserver/cs-npu/introduction/index.txt

Overview
NPU Cloud Servers, like Shared-CPU Cloud Servers, are managed through the Ermes web panel and the ECS API service. This means that all functions available for Shared-CPU Cloud Servers via the panel are also available for NPU instances.

Viewing NPU Instances
To view NPU instances, go to:
`Compute → NPU servers`
or navigate to: **Cloud Server NPU**.
The page displayed will be similar to the view for Shared-CPU Cloud Servers.

Creating a New NPU Instance
To create a new NPU Server, click **NEW SERVER** on the NPU Servers overview page or go directly to: **NPU Configurator**.
The Configurator will open.
The configuration process is identical to that for creating Shared-CPU Cloud Servers, with the only difference being the plan selection: in addition to choosing a plan, you can also configure the number of NPUs required. To do this, click on the NPU count and select the desired value from the available options.
To continue the configuration, you can follow: **Configure a Shared CPU**.

==

DOC: hosting/cloudserver/cs-shared/action/index.txt

Actions
Actions are very useful for monitoring the status of a procedure such as:

- Server creation
- Server deletion

- Template creation
- Template deletion
- Server power-on or power-off
- Console creation

For all these operations, you need to monitor the procedure’s status over time, which can be done by accessing:

`Compute → Actions`

or by going to **Actions**.

The page that opens will be similar to the following:

In the actions list (1) are all the actions that fall within the interval shown in the lower-left corner.

Navigating Actions
To move between different pages, use the navigation buttons in the lower-right corner (2):

- **Next** allows you to advance up to 20 actions
- **Previous** allows you to go back up to 20 actions
- **Recent** shows the last 20 received actions

Filtering Actions
It is also possible to filter actions by Server or Resource name; to do this you can use the search bar at the top right (n. 3) and then click the magnifying glass or press Enter.

==

DOC: hosting/cloudserver/cs-shared/api-token/index.txt

API Token
API Tokens are useful tools for integrating your remote services with our REST API quickly and easily.
To fully understand what API Tokens are and what they are used for, you need to explore User Management and API Tokens so you can have an overview of the API functionalities available in our services.
Although the use of API Tokens is reserved exclusively to the API, their creation and management can also be performed through the web panel.
To view the API Tokens you need to click on:
Compute → API Token
or go to API Token.
A page similar to the following will be displayed:

Add a Token
The procedure to add a Token is very simple: just click on **NEW TOKEN** and assign a name to the new Token:

After confirming, the name of the Token will be shown and the Token's content will be displayed as the result:

Once the token is generated it can be copied and used in any request to the ECS services API.

Attention
ECS services do not force token regeneration, but it is strongly recommended to do so over long periods for security reasons.

Remove a Token
To remove and disable a Token you need to click on **DESTROY**; you will be asked to confirm the token's destruction, and once confirmed the token will be unusable on the ECS services.

==

DOC: hosting/cloudserver/cs-shared/billing/index.txt

Consumption
It is possible to view the consumption of ECS Services directly from the WEB panel; to do this just click on **Consumption** in the left-hand menu of the panel.
A page similar to the following will open:

Both Server and Template consumptions are shown; each row represents the consumption of that resource for the specific month, including any reductions.

View past consumptions
It is also possible to view consumptions for previous months by selecting the desired year and month (n.1) and then clicking **CHANGE MONTH** (n.2).
After loading, the consumptions for the selected month will be displayed.

Export consumptions to CSV
It is also possible to export this data in CSV format; to do so you need to choose which resource you want to export—server or template—and click **EXPORT CSV** (n.3).

Attention
Note: if you try to export data while another month is still loading, the currently active month may be exported. To avoid this, wait until the loading of the month of interest has finished.

==

DOC: hosting/cloudserver/cs-shared/cloud-script/index.txt

Cloud Script

Cloud Scripts are one of the most powerful features of ECS services; they allow you to customize your Server with scripts written by the user.
In addition to temporary scripts, our services also expose a personal script library, making it possible to create reusable and customizable scripts.
To access this library you need to go to:
Compute → Cloud Script
or navigate to Cloud Script.
A page similar to the following will be shown:

Create a new Script
To create a new Script you need to click on the empty slot with the "+" symbol and enter a name for the Script.
You can have up to 20 personal Scripts.

The Web editor
To open a Script in the Web editor simply click on it; the following IDE will open in a new page:

In addition to editing the Script, the Web IDE provides several possible actions:

* You can save the Script at any time by clicking **SAVE** (n. 2) or using the keyboard shortcut **CTRL+R**.
* You can also modify both the Script title and the Language using the fields at the top left (n. 1).

Quick actions and Delete a Script
It is also possible to perform quick actions directly from the main page of your library; to do this click the three dots next to the Script name:

From here you can rename a Script, open it in the Web IDE, or delete the Script.
Once a Script is deleted its content will be unrecoverable.

==========

DOC: hosting/cloudserver/cs-shared/configurator/index.txt

Configurator
The configurator allows you to configure and create new Servers according to your needs.
To access the configurator you need to click on:
Compute → Shared CPU servers → NEW SERVER
or go to Shared CPU Configurator.
After doing so, the configuration page will open and guide you through the creation of the new virtual Server.

Info
The use of the configurator is very similar for other EC-compatible products with the Ermes web panel, such as:
For more information, visit the related pages.

Plan selection
The first step in Server configuration is to choose a plan:
A plan is simply a set of virtual resources (Core, RAM, Storage) assigned to a virtual Server; to select a plan just click on it and a summary will appear on the right.

Datacenter selection
... (the original text ends here)
Choosing a Datacenter
After that, you need to select a Datacenter from those available for the chosen plan:
The different datacenters show the city where the datacenter is located and its identification code.
Once a datacenter is selected, it will appear added to the summary on the right.

Choosing the Image
After selecting the datacenter where you want to place your virtual server, you can choose the corresponding image:
The image selection provides several components; in the center (no. 2) we have the list of sections containing different types of images:

- Standard Images
- Templates
- Cloud Images

Below that is the list of images available for the selected section (no. 1).
You can also use the search bar at the top (no. 3) to look for images in the current section.

Standard Images
Standard images are minimal operating systems without any configuration except basic users and network settings.
When using standard images, in addition to the operating system you can choose the specific version you want to use; to do this, click on the version under the OS name and change it as desired.

Templates
Templates appear differently from Standard images:
They have no version and represent a snapshot of an existing virtual server at a specific point in time. To create a new template you need to go to the Details of a server.
For more information on templates, see **Template**.

Cloud Images
Cloud Images are tools distinct from normal Standard images or Templates; they are custom-prepared by Seeweb’s expert engineers to solve specific problems.
They are specific versions of Standard images pre-configured with particular software, allowing users to create complex infrastructures

and/or install tools that normally require a cumbersome installation process, ready in a few seconds.
Once you have selected your Cloud Image (no. 1), you can view the image description in the panel located below the summary (no. 2).

Choosing Authentication
Another vital step in creating a server is choosing the authentication method. Currently, EC services support two types of authentication:

- A Temporary Password (OTP) (no. 1)
- SSH Keys (no. 2)

OTP Password
Using an OTP password is definitely the fastest way to configure your authentication; simply create your server and wait for the email containing the access credentials.

SSH Keys
Although OTP passwords are quick and convenient, sometimes a more secure method is needed for the initial authentication.
For example, if you are creating a server on behalf of someone else and want to avoid directly sharing OTP credentials.
To do this, our systems support authentication via SSH key: add your public key to our systems, choose the authentication type, and then select the key you wish to use.

Info
For more information on creating and managing SSH keys, visit: **SSH Keys**

Choosing Technical Support
Before proceeding to server creation, it is highly recommended to select the appropriate level of support you want to dedicate to your server.

Info
Support can also be added to an already active server; to do this or to learn more about technical support, see: **Add Technical Support**

Advanced Configurations
After completing all the steps above, you can choose the server name and proceed with its creation.
If you want deeper customization of your system, you need to open the **Advanced Configurations** section:
Here you will find several optional configuration options that can provide a higher degree of personalization for your virtual server.

Using a Reserved Period
Among the advanced configurations, Reserved Periods play a very interesting role:
Reserved periods guarantee resource capacity and cost savings over time, reducing maintenance expenses and locking resources for a defined period.
If you want to remove any reserved period from your server, click **Do not reserve** (no. 1).

Info
You can subscribe to a reserved period even for existing servers; to do this or to learn more about reserved periods, visit: **Reserved Periods**

Using a Cloud Script
Cloud Scripts are one of the most powerful features of ECS services; they allow you to customize your server with scripts you write yourself.
To do this, the configurator uses two tools:

- Temporary Scripts
- Private Scripts

The Cloud Script interface looks like this:

To use Cloud Scripts, first enable the feature by clicking **Use Cloud Script** (no. 1).

Temporary Cloud Scripts
Temporary Scripts let you write your script directly while configuring the server.
They are recommended for simple, quick tasks that do not require more than 20 lines of code.
To use them, select **Temporary** in the script list (no. 2); the IDE on the right (no. 3) will become editable, and you can write your script.

Private Cloud Scripts
For more complex and especially repetitive tasks, Temporary Cloud Scripts are not the best tool. In those cases, our services provide the ability to use your own Private Scripts.
Creating a New Script
To create a new Script you must use your own Library; after its creation you will be able to select it in the script list (no. 2). At this point the IDE on the right (no. 3) will display the content of the selected script.
This tool allows you to reuse your Scripts and makes customizing a virtual Server simpler and more efficient.

Environment Variables
After selecting one of your private Scripts you will notice that the Environment Variables section becomes interactive:

Environment variables let you use your Cloud Scripts dynamically by changing the value of specific variables while leaving the rest of the script unchanged.
To do this you can write the script using classic Bash variables such as `"$NAME"` and then reference them in the Environment Variables section as shown in the previous image.

To create a new variable simply click “+” (no. 1), enter the variable name (no. 2) and finally the variable’s content.
The variables will then be initialized at the beginning of the script.

Choosing a Group
You can also assign your Server directly to a specific Group. To do this, select the desired group from the dedicated screen after having created it in **Groups**:

Info
You can also assign an existing Server to a group directly from the Server Details.

Using Isolation
Another interesting feature for the security of your virtual Server is **Isolation**.
Isolation forces two or more Servers to be placed on different physical hardware. To use it, go to the following screen and select the Servers you want to isolate:

Info
You can view a Server’s isolation status directly from the Server Details.

Choosing the Server Name
As the final step you need to select a name for your Server. If you do not provide a name, one will be automatically generated based on the image, the plan, the datacenter and the current date.
To assign a name to your Server, enter it in the box below:

Summary
Before proceeding with Server creation it is always recommended to review the summary and be sure you are creating the desired Server:
Then simply click **“CREATE SERVER”** to start the creation process.

Tip
You can monitor the creation status directly from **Actions**.

DOC: hosting/cloudserver/cs-shared/dashboard/index.txt
The Dashboard
The dashboard is the entry point for the Shared CPU, High Memory, GPU and NPU Cloud Server service.
Our dashboard is designed to give you an overview of the status of your Servers and the latest news from the Seeweb ecosystem.
From here you can:

1. View your Servers (no. 1), sorted by type together with their resources.
2. See the latest communications from the Seeweb staff (no. 2).
3. See the Reserved Periods that are about to expire (no. 3).
4. Create a new Server directly by clicking the **‘NEW’** button at the top of the screen (no. 4).

DOC: hosting/cloudserver/cs-shared/groups/index.txt

Groups
Groups are a useful feature for grouping various servers.
They can be accessed directly from:

- **Compute → Groups**
- Or by going to **Groups**

The page will look similar to the following:

Creating a New Group
To create a new Group click **“NEW GROUP”** (no. 1) and enter a description for it.
Once created you can add Servers to it from the Configurator or from the Server Details.

Removing a Server from a Group
To remove a single Server from a group click **“REMOVE”** (no. 3) or go to the Server Details and select **“no group”**.

Deleting a Group
To completely delete a group click the trash can icon (no. 2) and confirm the removal.

DOC: hosting/cloudserver/cs-shared/introduction/index.txt
Overview
Shared CPU Cloud Server is the on-demand cloud service ideal for testing, development, staging, and for anyone who needs to create and destroy cloud machines with consumption-based billing, quickly and at optimized costs.
It allows full, autonomous management of your Servers via our simple Web Panel or through our API service, making it easy for both users and developers to use and integrate our services.

To access the web panel you must authenticate at the following address **Cloudcenter**, where you will be taken to the login page:
Enter your credentials there to be redirected to the management web panel.

DOC: hosting/cloudserver/cs-shared/logs/index.txt
Logs
Logs, together with Actions, aim to record all operations performed on the Shared CPU Cloud Server services.
Unlike Actions, which focus on whether the requested operation succeeded or failed, logs concentrate on the executor of the action and its origin.

Logs can be viewed by navigating to:

- **Compute → Logs**
- Or by going to **Logs**

The logs page appears as follows:
The navigation works similarly to the Actions page, but unlike it, clicking on the log **'ID'** opens the following page:
From here you can view all the information that our systems were able to obtain from a given request, allowing our users to get all the necessary details to identify the actor of an operation.

Info
For IP geolocation the open-source service **hostip.info** is used; this may mean that your IP might not be geolocatable at the time of the action.

==========

DOC: hosting/cloudserver/cs-shared/networks/index.txt

Networks
Networks are important tools that allow you to connect your virtual servers through a secure, private, and isolated channel in a simple and abstracted way.

Add a Network
To use networks you first need to create one; to do this go to the **"Networks"** section or click on **Networks**.
A page similar to the following will appear:
To create a network you must click **"ADD A NETWORK"**.
Once the network is created you can attach servers from its interface management page.

Analyze a Network
From this page you can also analyze a specific network; this operation lets you see all servers connected to it. To do so, click the scan icon on the row of the desired network.
The result will look like this:

Remove a Network
Obviously you can also remove a network; to do this click **"DESTROY"** on the row of the desired network.
If there are servers connected to the network at the time of deletion, the operation will result in an error and an automatic scan will be started to identify the instances that “block” the request.

==========

DOC: hosting/cloudserver/cs-shared/server/index.txt

Servers
Servers are the central element of the Ermes panel, supporting both Share-CPU servers and High-Memory, GPU, and NPU servers.

This page deals only with Shared-CPU servers.

View Servers
You can view your servers by accessing the corresponding page; to do this use the left sidebar and click:

```
Compute -> Shared CPU Servers
```

The following screen will be visible:
In the server list (item 2) all of the user's servers will be listed; to open a server's details simply click its name.
This operation is possible only for active servers.

Search a Server
When using the view pages the search bar becomes available, allowing you to filter servers by name or ID.

Best Practice
Always keep this functionality in mind when choosing a name for a new server and use a particularly descriptive label, e.g.:

- server-web-01
- db-fr-1
- db-fr-2

Or highlight characteristic attributes of the server such as the plan, datacenter, or operating system:

- eCS6-01
- debian-12-01
- eCS6-it-fr2-debian-12

This makes it much easier to find them quickly.

Compressed View
There are two display modes and you can switch between them at any time:

- **Compressed view** (automatic when there are more than 3 servers)
- **Expanded view** (automatic when there are 3 or fewer servers)

Switch by clicking the table expansion button (item 4).
The compressed view appears as described earlier.

In the compressed view you can get an idea of the server thanks to the descriptive icons placed on the row (item 6). To interpret these icons, follow the legend:

- --> The server is on a reserved plan
- --> Unmanaged Support
- --> Basic Support
- --> Global Support

- --> Proactive Support

Interact with the Server
In the compressed view you can interact with servers by opening the dropdown menu via the down-arrow (item 5). This opens a menu that lets you rename the server, open the console, or delete the server.

Expanded View
The expanded view provides more information about the server without needing to open the server details; it is recommended after searching for a server or when the number of servers is fewer than 3.
The expanded view appears as follows:

Interact with the Server
Even in the expanded view you can interact with the server and perform actions such as renaming the server or opening the console.

Create a New Server
To create a new server, open the configurator by clicking **"NEW SERVER"** (item 3).

==

DOC: hosting/cloudserver/cs-shared/server-details/index.txt

Server Details
The details page provides all significant information about a server and also allows you to perform all maintenance activities for that server, such as managing its state and various settings.
You can access the details page by clicking the server’s name from the view page.
The details page is divided into several sections and appears as follows:

Control Panel
At the top is the control panel, the only fixed element across sections; it contains various pieces of information about the server and the most generic actions. In detail it can be broken down as follows:
In the top-left corner (no. 1) you have the Server name and a link to return to the Server list; in the top-right corner (no. 2) you see the Server status and management actions, in this order:
- Power on/off button
- Rename operation
- Console

At the bottom you find the Server information summary, which includes:
- Server name
- Operating system
- Plan
- Core, RAM, Disk resources and, if present, GPUs
- Corresponding IPv4 and IPv6 addresses
- Server identifier (to be provided in case of platform malfunctions)
- The datacenter where the Server is located

- Creation date
- Finally, the Server version

Manage the status
Clicking the Power On or Power Off button lets you change your Server’s state. After clicking, a confirmation form appears; once confirmed, the requested operation is sent to the virtual Server.

Rename a Server
You can also rename your Server from here. Simply click the **RENAME** button and enter the desired name. Blank spaces are not allowed and will automatically be replaced with “-”.

Open the console
A remote console to the Server can be opened. The console view uses the well-known open-source library **noVNC**. Click **CONSOLE**, and a new window will appear that automatically attempts to connect to the virtual Server.

Note
Servers of type “v1” are being deprecated and do not support opening remote consoles.

The Resources
Immediately below the panel are several sections, including **RESOURCES**. This section tracks Server metrics and displays performance graphs, allowing a quick analysis of the machines’ state. It shows:
- CPU usage graph
- RAM usage graph
- Network traffic graph

Note
Metrics are not available for “v1” Servers or for the “eCSGPU8” plan.

Templates
The Templates section allows creation of templates from existing Servers. It appears as follows:

Creating a template is very simple: click **CREATE A TEMPLATE**, enter a description, and a template-creation request will be sent to the remote Server.

Note

Template creation is not supported for “v1” Servers or for the “eCSGPU8” plan.

Snapshots
Snapshots are copies of your virtual Server. Like Templates, they retain all data of your machine but differ in the following ways:
- You can restore a Server’s state from a Snapshot.
- A Snapshot cannot be used to create a new machine.
- A Snapshot can be converted into a Template, but not vice-versa.

Protected snapshots
Snapshots are divided into protected snapshots (indicated by a lock) and unprotected ones, with the following characteristics:

Manage snapshots
The Snapshots section looks like this:

From this screen you can perform all Snapshot management operations.

Create a Snapshot
To create a Snapshot, click **CREATE SNAPSHOT** in the top-right corner. You will be asked for a description and to confirm the creation process. The operation will not affect the current use of the Server, but it will disable the start of new actions on the Server until it completes.

Restore a Snapshot
If you want to roll back to a previous state, select the desired snapshot from the chart and click **RESTORE**. As shown, this operation requires a full machine reboot and may take several minutes. It is therefore advisable to run it only at appropriate times.

Clone into Template
Cloning into a template generates an independent Template from the Snapshot, allowing you to create new Servers from it. Click **CLONE INTO TEMPLATE** and provide a description for the new image.

Protect a Snapshot
Protecting a Snapshot excludes it from the “Maximum Retention” rule configured for Periodic Snapshots, so periodic snapshots cannot delete it. Click **PROTECT** and confirm the operation.

Rename and Delete a Snapshot
Renaming or deleting a Snapshot is straightforward: select the desired Snapshot and use the **RENAME** and **DELETE** commands respectively.

Periodic Snapshots
Snapshots can also be scheduled to provide weekly protection for the Server. To do this, configure a cron job on the relevant Server.

Currently, only weekly schedules are available with the following parameters:

Info
“Maximum retention” indicates the maximum number of unprotected snapshots allowed. If creating a new Snapshot would exceed this number, the oldest unprotected Snapshot is deleted first.

Configure Periodic Snapshots
To configure Periodic Snapshots, click the **SCHEDULE** option next to the creation button. The following menu opens:

From here you can select the day of the week on which the Snapshot will be taken and the desired maximum retention. After confirming, the updated menu is shown, allowing you to modify the settings over time.

Note
The schedule attribute can be rescheduled within the time window of 00:00 – 06:00 UTC.

Interfaces
The network section allows you to attach your Server to a private network.
To do this, you need to create at least one private network.
The interfaces are actually network devices added to the virtual server and have the following characteristics:

- an interface requires at least one active VLAN to exist;
- deleting all VLANs will cause the interface to be destroyed;
- creating and deleting an interface will automatically reboot the server.

Even though at least one VLAN is required to create an interface, you do not need to know this technology to use the private networks of our services.
When an interface is added, our web panel will automatically create a base VLAN to allow the network to work.

The basic configuration will bring us to the following state:

To deepen the topic of VLANs and get the most out of private networks, it is advisable to start with the following papers:

The section will appear as follows:

Add an Interface
From this section you can add as many interfaces as there are networks you have created. To do this, click **“ADD AN INTERFACE”** (item 1) and select the desired network.
Creation will cause the machine to reboot, and once the operation is complete the MAC address of the newly created interface will be displayed.

Manage VLANs
From this section you can also manage all VLANs of an interface.
To do this, use the commands on the right.
Above the text field are special commands, made available to allow faster management of our VLANs.

You can therefore:

- select a single VLAN: `"10"` (if you want to select VLAN 10)
- select a range of VLANs: `"10-20"` (if you want to select all VLANs from 10 to 20)
- send multiple commands in succession: `"10,20-30"` (if you want to select VLAN 10 and all VLANs from 20 to 30)

You can also type `*` to automatically select all VLANs from 1 to 4094, or `!` to insert all VLANs not yet created.

Create a VLAN
To create an interface you must enter the selection command in the appropriate field and click **“CREATE VLANS”** (item 2).
The VLAN will be created and become available when the status (the circle on the left) turns green.

Attention
Attempting to create a VLAN with the same ID as a VLAN that has just been deleted may cause an error.
To avoid this, wait until the deletion action is completed.

Remove a VLAN
You can remove one or more VLANs using the commands described above and clicking **“REMOVE VLANS”** (item 4).

Attention
To remove or modify a VLAN, it must be in a stable state, indicated by the dot next to the VLAN ID.
Attempting to modify a VLAN in an unstable state will result in an error.

Modify the PVID
From here you can also select a new PVID; simply specify an already-created VLAN and click **“PVID”** (item 3) to start the update process.

Configure an Interface on the Server
Although all infrastructural operations are performed by our services, a simple manual configuration is required to allow our instances to communicate.
This gives the end-user full control of their network with the resulting flexibility.

You therefore need to log into your servers and configure the network address you want to use for communication.
You can follow this quick example:

The example assumes you want to configure the network `192.168.1.0/24` (the network choice is entirely up to the user).
The example refers to the following servers with the listed IP addresses and MAC addresses, but it can be replicated with any server:

Both servers are connected to the network: **“net000026”**

1. First, access the console of these servers via the web panel or via SSH.
2. Then determine which interface is attached to the network by comparing the MAC address shown in the panel with the MAC addresses inside each server:

Taking the server **“test-1”** as an example:

```bash
ip link
```

You will get an output similar to:

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT group default qlen 1000
    link/ether 10:00:ac:11:00:14 brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/ether 52:54:00:b9:27:40 brd ff:ff:ff:ff:ff:ff
```

By comparing the MAC addresses, you can see that interface **eth1** has the same MAC address shown in the panel.
You can then configure **eth1** to communicate with server **test-2**:

Bring up the interface

```bash
sudo ip link set dev eth1 up
```

Add a route for the 192.168.1.0/24 network via eth1

```bash
sudo ip route add 192.168.1.0/24 dev eth1
```

Assign an IP address to the interface (for test-1 we use 192.168.1.1)

```bash
sudo ip address add 192.168.1.1/24 dev eth1
```

```

Now perform the same steps on **test-2**:

```bash
sudo ip link set dev eth1 up
sudo ip route add 192.168.1.0/24 dev eth1
sudo ip address add 192.168.1.2/24 dev eth1
```

Finally, the two servers can communicate securely on their private network. You can test this with the following command (run on **test-1**):

```bash
ping 192.168.1.2 -c 5
```

You will get the following result:

```
PING 192.168.1.2 (192.168.1.2) 56(84) bytes of data.
64 bytes from 192.168.1.2: icmp_seq=1 ttl=64 time=0.369 ms
```
64 bytes from 192.168.1.2: icmp_seq=2 ttl=64 time=0.464 ms
64 bytes from 192.168.1.2: icmp_seq=3 ttl=64 time=0.419 ms
64 bytes from 192.168.1.2: icmp_seq=4 ttl=64 time=0.453 ms
64 bytes from 192.168.1.2: icmp_seq=5 ttl=64 time=0.455 ms

--- 192.168.1.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 11ms
rtt min/avg/max/mdev = 0.369/0.432/0.464/0.035 ms

Attention
These commands may not make the network configuration persistent; in other words, the configuration could be lost on a machine reboot, especially if tools such as a network-manager are used. To avoid this, you need to configure the network devices persistently. Persistent configuration varies from distribution to distribution and is not covered by this guide.

Settings
The Settings section is the most advanced part of the Details page; it contains several subsections, each with specific objectives, and appears as follows:

Reserved Periods
One of the most interesting features in Settings is certainly the Reserved Periods:

Reserved periods guarantee resource capacity and time savings, reducing maintenance costs and locking resources for a defined period.

Add a Reserved Period
```

To do this, click **“ADD A RESERVED PERIOD NOW”** and the following menu will appear:

Here you can select the time span during which to reserve the specific Server. Doing so will make it impossible to delete the Server until the period expires. The discount percentage varies based on the reserved period and the plan.

The Group
You can assign a Server to a Group from the “Group” subsection.

First, create the desired group. Then you can add a Server to the group from the following menu:

Isolation
Isolation is an advanced feature that forces two Servers to be placed on different physical hardware; it can be used directly within the configurator.

From this page you can also view the isolation status of the specific Server:

On the left is the Server we are analyzing, and on the right are the other Servers from which it is isolated.

Technical Support
Activating a technical support plan is highly recommended because it allows you to receive assistance from Seeweb’s expert engineers for your specific Server.

Support plans for Servers are subject to a monthly pay-per-use fee; destroying the Server will not stop billing for support in the current usage month.

Available support plans:

- The **Unmanaged** plan is enabled by default and provides the minimum support guaranteed by Seeweb technicians.
- **Basic** and **Global** plans offer greater benefits and higher priority for your requests.
- The **Proactive** plan is the most comprehensive support service we offer and guarantees the highest priority for your requests. For more information, see **Proactive**.

Destroy a Server
Finally, you can also destroy a Server from the last section of Settings:

The operation is very simple—just click the trash can icon and confirm the deletion.

DOC: hosting/cloudserver/cs-shared/ssh-key/index.txt
SSH Keys

Using SSH keys is strongly recommended. Although you can use temporary passwords (OTP) to access newly created Servers, it is much safer to use your own SSH key.

To access the SSH key management panel, open the user-management dropdown menu and select **“SSH Keys”**.

You will then see the following page:

Here you can:

- View SSH keys (1)
- Add a new SSH key (2)
- Remove an SSH key (3)

Add a Key
Before you can use an SSH key, you must register your public key with our services.

Click **“ADD”** (2); the following form will appear:

You need to generate the SSH key (2) and paste the public-key content into the appropriate fields (1).

Instructions for generating the key are provided on the right side of the form (2).

Once generated, the key can be used when creating any Server.

Attention
Remember: registration will fail if:

- The SSH key name is not unique for the user
- The SSH key does not follow the proper format

Remove an SSH Key
To remove an SSH key, click **“REMOVE”** (3). A confirmation form will appear; after confirming, the key will be completely deleted from our systems.

DOC: hosting/cloudserver/cs-shared/templates/index.txt
Template
Templates, unlike Standard Images, are created by the user and are private.

They allow you to create images from servers already in use while preserving:

- Users and their passwords
- Installed software
- Every file on the system

They do **not** preserve:

- Network interfaces
- Associated IPs and MAC addresses

To create a template, go to the Details page of the Server from which you want to generate the Template.

View Templates
To view your templates, simply go to the corresponding page by clicking:

Compute → Templates

or by navigating to **Templates**.

The page will look similar to the following:

Remove a Template

To remove a Template, simply click on **"DELETE"** on the row of the Template you want to delete; you will then be asked to confirm the operation, and once done all content of the Template will be permanently removed.

==
========

DOC: hosting/cloudserver/cs-shared/terraform/index.txt

Overview
Terraform is a tool that allows you to create cloud and on-prem resources with human-readable configuration files that can be reused and shared. These files can then be used to perform provisioning and manage the infrastructure throughout its entire lifecycle.

How does it work?
Terraform creates and manages resources on cloud platforms and other services through their APIs. Providers enable Terraform to work with platforms or services that have accessible APIs.

The Terraform workflow consists of three steps:

* **Create**: you can define resources, which may span multiple cloud providers and services. For example, you can create a configuration to deploy an application on a virtual machine, in a Virtual Private Cloud (VPC).
* **Plan**: Terraform creates an execution plan describing the infrastructure that will be created, updated, or destroyed based on the existing infrastructure and your configuration.
* **Apply**: after approval, Terraform carries out the operations in the correct order, respecting every resource dependency. For example, if you update the properties of a VPC by changing the number of virtual machines

inside it, Terraform will recreate the VPC before scaling the virtual machines.

Terraform Provider for Seeweb

Configuring the Provider
```bash
# Clone the repository into:
$GOPATH/src/github.com/Seeweb/terraform-provider-seeweb

$ mkdir -p $GOPATH/src/github.com/Seeweb; cd
$GOPATH/src/github.com/Seeweb
$ git clone git@github.com:Seeweb/terraform-provider-seeweb
```

Enter the provider folder and configure it:
```bash
$ cd $GOPATH/src/github.com/Seeweb/terraform-provider-seeweb
$ export GOPRIVATE=github.com/Seeweb/*   # This step is only necessary if
the modules are kept private
$ make build
```

Using the locally configured Provider
To test a local version of a Terraform Provider, run these commands:
```bash
go build -o terraform-provider-seeweb
# The next location might be different on your machine, so check first
mv terraform-provider-seeweb
~/.terraform.d/plugins/registry.terraform.io/hashicorp/seeweb/0.0.1/darwi
n_arm64
```

You can then request the Seeweb provider, as in the example below:

Example
```hcl
provider "seeweb" {}   # Expecting Seeweb auth token in env var
$SEEWEB_TOKEN

resource "seeweb_server" "testacc" {
  plan     = "ECS1"
  location = "it-fr2"
  image    = "centos-7"
  notes    = "created with Terraform"
}
```

Testing
To test the provider, simply run `make test`
```bash
$ make test
```

To run the full acceptance test suite, execute `make testacc`.
* The tests require the environment variable `SEEWEB_TOKEN` to be set to a valid API token.

Note
Acceptance tests create real resources and can be costly to run.
```bash
$ make testacc
```

Run a specific subset of tests by name using the `TESTARGS="-run TestName"` option, which runs all test functions containing “TestName” in their name.
```bash
$ make testacc TESTARGS="-run TestAccSeewebServer"
```

Reference
https://registry.terraform.io/providers/Seeweb/seeweb/latest/docs

==

DOC: hosting/cloudserver/cs-shared/user-management/index.txt

User Management
User management is one of the simplest operations you can perform from any part of the web panel via the top bar: just click on your username to open the dropdown menu related to user management.

From there you can:

- change your credentials;
- configure your SSH keys;
- log out of the panel.

Changing Your Credentials
To change your credentials, click on **“Change Password”** within the account management section; the following form will appear:

In the central form (item 1) you can enter the new login credentials. The “save credentials” button will be enabled only if the requirements shown in the bottom-right corner (item 2) are fully satisfied.

Why Should I Log Out?
Even though our services use temporary JWT tokens, logging out is extremely important for securing your local application environment. Doing so makes our systems “forget” who you are, forcing any malicious party that may have taken over your terminal to re-authenticate with credentials.

==

DOC: hosting/cloudserver/rest-api/API-Endpoints/Account/index.txt
---

Account

Various operations can be performed via the API for managing user accounts, including login and retrieving account information.

Login
User-related operations are processed through our API, so a JWT token must be generated to allow the user to identify themselves to our services. The following endpoint generates an access token from a username and password.

HTTP Request
```http
POST /ecs/v2/login
{
  "username": "foo",
  "password": "foo-s3cr3t-passw0rd"
}
```

Response
```json
{
  "token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJtc2ciOiJXcm9uZyBQbGFjZSBkdWRlPyJ
9.vBb9pIUYaxiGtO-7HY6hzBXH-q7L0GJS8stz2VYqb90",
  "expire": 1728560902,
  "status": "ok"
}
```

Info
The generated token can be used as a standard Bearer token until it expires.

```bash
curl --request GET 'https://api.seeweb.it/ecs/v2/' \
```
--header 'Authorization: Your-JWT-Token' \
Account Information
User information can be retrieved via the following endpoint:
HTTP Request
GET /ecs/v2/account
{
  "username": "foo",
  "email": "foo@seeweb.it",
  "reserved_available": true, # The user can use the reserved plan
  "offline_available": false, # The user receives an offline discount
  "ecs_special_discount": 12, # if > 0 the user already has a special discount on standard ECS plans

```
  "ecshm_special_discount": 0,  # if == 0 the user has no special
discounts on high-memory plans
  "ecsgpu_special_discount": -1 # if < 0 the user has a special discount
on GPU plans
}
```

Change Password

You can change a user's password directly via the API at the following endpoint:

HTTP Request

```
POST /ecs/v2/account
{
  "password": "my-new-supers3cr3t-passw0rd"
}
{
  "status": "ok"
}
```

API Token

TL;DR:

API tokens are used to integrate ECS services with your applications without, unlike JWT tokens, any expiration.

To use it, just add it as a header:

```bash
curl --location --request GET 'https://api.seeweb.it' \
  --header 'X-APITOKEN: *your API token*'
```

Read on:

JWT tokens are very useful for validating a user's actions for short periods, but what can you do if you need to connect ECS systems to an automated service?

To solve this problem we introduced API tokens, long-lived tokens that allow integration with our systems without the need to generate a JWT for each request.

Warning

ECS services do not force token regeneration, but it is strongly recommended to do so over long periods for security reasons.

Create an API Token

To create an API token you can simply use the following endpoint:

HTTP Request

```
POST /ecs/v2/keys
{
  "notes": "foo-rest-api-token-label" # a name used to identify the token
}
{
  "status": "ok",
  "token":
"8yKnCbmPMyAgeaxjhzhdWc7WtsI9Y0XVvuavy0m9E1co6S0epK3e0DPqRjzpBDyGBxFeeFdY
KFgpPfWZaTdNu2K1hdhcvviarI4biDNrybmQUY4LLbHtM6C5GsW2EIhc"
}
```

Info

After creating the token, you can simply add it to the "X-APITOKEN" header as follows:

```bash
curl --location --request GET 'https://api.seeweb.it' \
  --header 'X-APITOKEN: *your API token*'
```

```
**Warning**
This token is shown only immediately after creation; if the token is lost
or compromised it is strongly recommended to destroy it and generate a
new one.
List API Tokens
Obviously ECS is not authorized to share the token's content, but it can
share the labels and the code associated with each token so they can be
deleted:
HTTP Request
GET /ecs/v2/keys
[
  {
    "id": 0,
    "notes": "foo-rest-api-token-label-1",
    "created_at": "2024-10-10 10:36..."
  },
  {
    "id": 1,
    "notes": "foo-rest-api-token-label-2",
    "created_at": "2024-10-10 10:37..."
  }
]
Rename an API Token
To change the label of your API token you can simply use the following
request:
HTTP Request
PUT /ecs/v2/keys/{token-id}
{
  "notes": "my-new-label"
}
{
  "status": "ok"
}
Delete an API Token
To delete an API token the following request is available:
HTTP Request
DELETE /ecs/v2/keys/{token-id}
{
  "status": "ok"
}

================================================================================

DOC: hosting/cloudserver/rest-api/API-Endpoints/Action/index.txt
--------------------------------------------------------------------

Actions
Actions represent the state of a request over time; they are very
important for tracking all active, completed, or failed asynchronous
requests and their current status.
Latest Actions
To retrieve the latest 50 actions a simple request is available:
```

HTTP Request
GET /ecs/v2/actions
HTTP Response

```
{
  "status": "ok",
  "actions": [
    {
      "id": 709, // action id
      "status": "completed", // current status of the action ['in-
progress', 'completed', 'failed']
      "user": "foo",
      "created_at": "2024-10-19T15:08:55.887799+00:00",
      "started_at": "2024-10-19T15:08:55.887284+00:00",
      "completed_at": "2024-10-19T15:09:56.067592+00:00",
      "resource": "ec200285", // which object the action works on
      "resource_type": "ECS", // resource type
      "type": "delete_server", // action type
      "progress": 100 // progress
    },
    {
      "id": 708,
      "status": "completed",
      "user": "foo",
      "created_at": "2024-10-18T15:08:55.887799+00:00",
      "started_at": "2024-10-18T15:08:55.887284+00:00",
      "completed_at": "2024-10-18T15:09:56.067592+00:00",
      "resource": "ec200285",
      "resource_type": "ECS",
      "type": "new_server",
      "progress": 100
    },
    ...
  ],
  "total_actions": 754 // all available actions
}
```

Search Among Actions
To perform targeted searches among actions our request exposes several parameters that allow pagination of results and searching for actions by resource.
Pagination
To paginate results the parameters `start` and `length` are available, which define respectively:
- From which action to start pagination
- How many actions to return
For example, to obtain 20 actions while skipping the latest 40 actions:
HTTP Request
GET /ecs/v2/actions?start=40&length=20
Info
Remember actions are always ordered by most recent first.
Search by Resource
Besides pagination, another important feature for actions is searching by resource; this can be done using the `resource` parameter.
Suppose we want to search for all actions concerning `ec000001`.
HTTP Request

GET /ecs/v2/actions?resource=ec000001
Info
If both pagination and resource search are used, pagination takes precedence over the resource filter.
If I execute the following request:
GET /ecs/v2/actions?resource=ec000001&start=40&length=20
I will get 20 actions concerning `ec000001` after skipping the first 40; pagination is therefore applied after extracting the actions related to `ec000001`.
A Specific Action

Obviously it is also possible to monitor a single action; this can be useful to verify whether a server was created or not.
To do this, simply use the following request:

HTTP Request
`GET /ecs/v2/actions/{actionid}`

HTTP Response
```json
{
  "status": "ok",
  "action": {
    "id": 709, //action id
    "status": "completed", //current status of the action ['in-progress', 'completed', 'failed'],
    "user": "foo",
    "created_at": "2024-10-19T15:08:55.887799+00:00",
    "started_at": "2024-10-19T15:08:55.887284+00:00",
    "completed_at": "2024-10-19T15:09:56.067592+00:00",
    "resource": "ec200285", //the object the action works on,
    "resource_type": "ECS", //resource type
    "type": "delete_server", //action type
    "progress": 100 //progress
  }
}
```

DOC: hosting/cloudserver/rest-api/API-Endpoints/Billing/index.txt

Consumption
Our APIs also expose platform consumption data, which is available in two formats:

- CSV
- JSON

Consumption data is divided by **Server** and by **Template**.

Server Consumption
To retrieve server consumption, the following endpoint is provided:

HTTP Request
`GET /ecs/v2/billing/servers/<year>/<month>`

Server Consumption CSV
To obtain consumption in CSV format, use the `type` query parameter:

HTTP Request
`GET /ecs/v2/billing/servers/<year>/<month>?type=csv`

You will receive a response of type `text/csv` containing the requested consumption data.

Server Consumption JSON
To obtain consumption in JSON format, use the `type` query parameter:

HTTP Request
`GET /ecs/v2/billing/servers/<year>/<month>?type=json`

HTTP Response
```json
[
  {
    "plan": "ECS1",
    "name": "ec200200",
    "group": null,
    "notes": "centos-ecs1-it-fr2-1713866700891",
    "status": "Booted",
    "year": 2024,
    "month": 4,
    "creation_date": "2024-04-23T10:06:22.973332+00:00",
    "deletion_date": null,
    "billed_hours": 182,
    "cost": 2.64,
    "summary": {
      "online_hour": 1,
      "online_price": 0.02,
      "offline_hour": 0,
      "offline_discount": 20,
      "offline_price": 0.0,
      "offline_reduction": 0.0,
      "reserved_discounts": [
        {
          "reserved_plan": 26,
          "reserved_hour": 181,
          "reserved_discount": 15,
          "reserved_price": 2.62,
          "reserved_reduction": 0.46,
          "reserved_start_date": "2024-04-23T10:06:22.973332+00:00",
          "reserved_end_date": "2024-10-23T10:06:22.973332+00:00"
        }
      ],
      "original_cost": 3.09
    }
  }
```

```
]
```

Template Consumption
To retrieve template consumption, the following endpoint is provided:

HTTP Request
`GET /ecs/v2/billing/templates/<year>/<month>`

Template Consumption CSV
To obtain consumption in CSV format, use the `type` query parameter:

HTTP Request
`GET /ecs/v2/billing/templates/<year>/<month>?type=csv`

You will receive a response of type `text/csv` containing the requested consumption data.

Template Consumption JSON
To obtain consumption in JSON format, use the `type` query parameter:

HTTP Request
`GET /ecs/v2/billing/templates/<year>/<month>?type=json`

HTTP Response
```json
[
  {
    "cost": 0.18,
    "name": "ei200112",
    "status": "Created",
    "creation_date": "2025-02-05T08:10:49.938995+00:00",
    "deletion_date": null,
    "notes": "MyDemo",
    "year": 2025,
    "month": 2,
    "hours": 55,
    "size_on_disk": 1510080
  }
]
```

Snapshot Consumption
To retrieve snapshot consumption, the following endpoint is provided:

HTTP Request
`GET /ecs/v2/billing/snapshots/<year>/<month>`

Snapshot Consumption CSV
To obtain consumption in CSV format, use the `type` query parameter:

HTTP Request
`GET /ecs/v2/billing/snapshots/<year>/<month>?type=csv`

You will receive a response of type `text/csv` containing the requested consumption data.

Snapshot Consumption JSON
To obtain consumption in JSON format, use the `type` query parameter:

HTTP Request
`GET /ecs/v2/billing/snapshots/<year>/<month>?type=json`

HTTP Response
```json
{
  "year": 2025,
  "month": 2,
  "total_cost": 3.7,
  "rows_count": 37,
  "snapshots": [
    {
      "snapshot": 58,
      "snapshot_name": "ec200404-SNP-6",
      "snapshot_status": "DD",
      "snapshot_status_label": "Deleted",
      "snapshot_description": "periodic snapshot - 2025-01-
31T12:00:00.104930+00:00",
      "snapshot_created_at": "2025-01-31T12:00:00.129801+00:00",
      "snapshot_deleted_at": "2025-02-03T16:36:55.938131+00:00",
      "size_on_disk": 1571400,
      "hours": 65,
      "final_cost": 0.1
    }
  ]
}
```

DOC: hosting/cloudserver/rest-api/API-Endpoints/Cloud-Script/index.txt

Cloud Script
Cloud Scripts are an important feature that allows the configuration of a server during creation via a remote script.
In addition to temporary scripts, our services also expose a personal script library, making it possible to create reusable and customizable scripts.
Although library management is recommended through the web panel, it can also be performed via the API.

Info
Remember: the use of environment variables is only possible with scripts that come from the script library; it is not allowed with temporary scripts.

Adding a New Script
To add a new script, you can use the following request:

HTTP Request
```json
POST /ecs/v2/scripts
{
  "title": "my-script",
  "content": "#!/bin/bash\necho \"Hello, World!\" > /root/greet.txt",
  "windows": false //indicates whether the script is intended for Windows
machines
}
```

HTTP Response
```json
{
  "id": 0, //identifier
  "user": "foo",
  "title": "my-script",
  "content": "#!/bin/bash\necho \"Hello, World!\" > /root/greet.txt",
  "windows": false, //indicates whether the script is intended for
Windows machines
  "public": false, //indicates whether the script is public to other
users
  "category": null
}
```

You may encounter the following errors:

Retrieving Existing Scripts
To obtain all scripts belonging to the user, you can execute the
following request:
HTTP Request
GET /ecs/v2/scripts

HTTP Response
```json
{
  "status": "ok",
  "scripts": [
    {
      "id": 0, //identificativo
      "user": "foo",
      "title": "my-script",
      "content": "#!/bin/bash\necho \"Hello, World!\" > /root/greet.txt",
      "windows": false, //indica se lo script è destinato a macchine
windows
      "public": false, //indica se lo script è pubblico per altri utenti
      "category": null
    },
    {
      "id": 1, //identificativo
      "user": "foo",
      "title": "my-script-w",
```

```
      "content": "#ps1\necho \"Hello, World!\" >
C:\\Users\\Administrator\\Desktop\\greet.txt",
      "windows": true, //indica se lo script è destinato a macchine
windows
      "public": false, //indica se lo script è pubblico per altri utenti
      "category": null
    }
  ]
}
```

Get a Specific Script
To retrieve a particular script you can use the following request:

HTTP Request
GET /ecs/v2/scripts

HTTP Response
```json
{
  "status": "ok",
  "script": {
    "id": 0, //identificativo
    "user": "foo",
    "title": "my-script",
    "content": "#!/bin/bash\necho \"Hello, World!\" > /root/greet.txt",
    "windows": false, //indica se lo script è destinato a macchine
windows
    "public": false, //indica se lo script è pubblico per altri utenti
    "category": null
  }
}
```

Modify a Specific Script
You can modify an already-created script using the following request:

HTTP Request
PUT /ecs/v2/scripts/{scriptid}
```json
{
  "title": "my-script-renamed", //opzionale
  "content": "#!/bin/bash\necho \"Hello, Guys!\" > /root/greet.txt",
//opzionale
  "windows": false, //opzionale
}
```

HTTP Response
```json
{
  "status": "ok",
  "script": {
    "id": 0, //identificativo
```

```
    "user": "foo",
    "title": "my-script-renamed",
    "content": "#!/bin/bash\necho \"Hello, Guys!\" > /root/greet.txt",
    "windows": false, //indica se lo script è destinato a macchine
windows
    "public": false, //indica se lo script è pubblico per altri utenti
    "category": null
  }
}
```

You might encounter the following errors:

Delete a Script
You can remove scripts you no longer need with the following request:

HTTP Request
DELETE /ecs/v2/scripts/{scriptid}

DOC: hosting/cloudserver/rest-api/API-Endpoints/Group/index.txt
Groups
Groups are a visual feature usable from the Ermes panel to provide grouping of various servers; nevertheless they can be viewed and configured via the API.

Create a Group
To use groups you first need to create one. The following request is available:

HTTP Request
POST /ecs/v2/groups
```json
{
  "notes": "my-faboulus-group", //l'etichetta del tuo gruppo
  "password": "" //il campo password è deprecato ma è necessario nella
richiesta per questioni di retrocompatibilità
}
```

HTTP Response
```json
{
  "status": "ok",
  "group": {
    "id": 1,
    "name": "ei000001",
    "notes": "my-faboulus-group",
    "enabled": true
  }
}
```

You might receive the following errors:

Once a group is created you can add servers to it.

View Active Groups
You can view all groups using the following request:

HTTP Request
GET /ecs/v2/groups

HTTP Response
```json
{
  "status": "ok",
  "groups": [
    {
      "id": 1,
      "name": "ei000001",
      "notes": "my-faboulus-group",
      "enabled": true
    },
    {
      "id": 2,
      "name": "ei000002",
      "notes": "my-second-faboulus-group",
      "enabled": true
    }
  ]
}
```

View a Specific Group
To view a particular group the following request is available:

HTTP Request
GET /ecs/v2/groups/{groupid}

HTTP Response
```json
{
  "status": "ok",
  "group": {
    "id": 1,
    "name": "ei000001",
    "notes": "my-faboulus-group",
    "enabled": true
  }
}
```

Delete a Group
If you want to delete a group you can do so with the following request:

HTTP Request

DELETE /ecs/v2/groups/{groupid}

HTTP Response
```json
{
  "status": "ok"
}
```

You might receive the following errors:

DOC: hosting/cloudserver/rest-api/API-Endpoints/Image/index.txt
Images
Images are the operating systems that can be installed natively on our servers; unlike Templates they are public and available to everyone.

Get Base Images
To retrieve all active base images you can use the following request:

HTTP Request
GET /ecs/v2/images/basics

HTTP Response
```json
{
  "status": "ok",
  "images": [
    {
      "id": 1, //image id
      "name": "centos-7", //image code
      "creation_date": "2019-04-16T13:37:41+00:00",
      "active_flag": true,
      "status": "CD",
      "uuid": "ffe2a034-7a44-4f51-9ae8-17fb354793f6",
      "description": "CentOS Linux 7", //a public description of the specific version
      "notes": "Linux CentOS disk image", //an internal note
      "public": true,
      "cloud_image": false,
      "so_base": "centos", //base os
      "api_version": "v4",
      "version": "7" //base os version
    },
    {
      "id": 2,
      "name": "debian-9",
      "creation_date": "2019-04-18T12:34:08+00:00",
      "active_flag": true,
      "status": "CD",
      "uuid": "6708cdc4-0c8d-4ccf-83c7-ae39bccaeb42",
      "description": "Debian Linux 9",
      "notes": "Linux Debian disk image",
```

```
      "public": true,
      "cloud_image": false,
      "so_base": "debian",
      "api_version": "v4",
      "version": "9"
    },
    ...
  ]
}
```

Get Cloud Images
Cloud images are essentially customized base images similar to Cloud Scripts. Unlike Cloud Scripts they are usable at the image level and are tightly linked to the operating system they derive from.

To see the active cloud images you can use the following request:

HTTP Request
GET /ecs/v2/images/cloud-images

HTTP Response
```json
{
  "status": "ok",
  "images": [
    {
      "id": 40,
      "name": "sci-gpu-seeweb-ai-ubuntu-2204",
      "creation_date": "2024-06-24T10:44:15.494000+00:00",
      "active_flag": true,
      "status": "CD",
      "uuid": "2d41e9c4-280d-423b-8110-9d14173f601c",
      "description": "Python AI Toolkit",
      "notes": "Seeweb AI Toolkit - Ubuntu Linux 22.04 LTS",
      "public": true,
      ...
    }
  ]
}
```
```json
"cloud_image": true,
"so_base": "ubuntu",
"api_version": "v4",
"version": "2204"
},
{
"id": 41,
"name": "sci-gromacs-2022.2-ubuntu-2204",
"creation_date": "2024-06-24T10:44:41.206000+00:00",
"active_flag": true,
"status": "CD",
"uuid": "2d41e9c4-280d-423b-8110-9d14173f601c",
```

"description": "Gromacs 2022.2",
"notes": "Gromacs 2022.2 inside docker on Ubuntu Linux 22.04 LTS",
"public": true,
"cloud_image": true,
"so_base": "ubuntu",
"api_version": "v4",
"version": "2204"
},
]
}
```

---

## DOC: hosting/cloudserver/rest-api/API-Endpoints/Logs/index.txt

### Log
Logs together with Actions aim to record all operations performed on the Cloud Server Shared CPU services. Unlike Actions, which focus on whether the requested operation succeeded or failed, logs concentrate on the executor of the action and its origin.

#### Retrieve Logs
To retrieve logs or make queries among them, a simple request is available at the following endpoint:

**HTTP Request**
`GET /ecs/v2/logs`

**HTTP Response**
```json
{
  "total_logs": 477,
  "logs": [
    {
      "id": 480,
      "actor": "foo",
      "action": "Login",
      "severity": "info",
      "status": "success",
      "target_type": "User",
      "target": "foo",
      "created_at": "2025-01-17T08:35:42.521Z",
      "ip_address": "127.0.0.1",
      "user_agent": null,
      "url": "https://api.seeweb.it/ecs/v2/login",
      "location": null,
      "result_type": null,
      "result": null,
      "description": null,
      "extra_data": null
    },
    {
      "id": 479,

```
      "actor": "foo",
      "action": "Login",
      "severity": "info",
      "status": "success",
      "target_type": "User",
      "target": "foo",
      "created_at": "2025-01-17T08:31:45.196Z",
      "ip_address": "127.0.0.1",
      "user_agent": null,
      "url": "https://api.seeweb.it/ecs/v2/login",
      "location": null,
      "result_type": null,
      "result": null,
      "description": null,
      "extra_data": null
    }
  ]
}
```


Just like with Actions, this endpoint also supports the parameters:
- `start`
- `length`
- `resource`

For more information, see **Search Actions**.

Retrieve a Specific Log
You can also obtain a specific log by using its ID:

HTTP Request
`GET /ecs/v2/logs/{logid}`

HTTP Response

```json
{
  "id": 480,
  "actor": "foo",
  "action": "Login",
  "severity": "info",
  "status": "success",
  "target_type": "User",
  "target": "foo",
  "created_at": "2025-01-17T08:35:42.521Z",
  "ip_address": "127.0.0.1",
  "user_agent": null,
  "url": "https://api.seeweb.it/ecs/v2/login",
  "location": null,
  "result_type": null,
  "result": null,
  "description": null,
  "extra_data": null
}
```

DOC: hosting/cloudserver/rest-api/API-Endpoints/Manage-Servers/index.txt

Managing a Server
A server may require various management operations. To address these needs, our systems expose several requests for powering the server on, powering it off, and restoring it to a previous snapshot.

Power On a Server
To power on a server, use the following request:

HTTP Request
```http
POST /ecs/v2/servers/{servername}/actions
{
  "type": "power_on"
}
```

HTTP Response
```json
{
  "id": 709,
  "status": "in-progress",
  "user": "foo",
  "created_at": "2024-10-19T15:08:55.887799+00:00",
  "started_at": "",
  "completed_at": "",
  "resource": "ec200001",
  "resource_type": "ECS",
  "type": "power_on",
  "progress": 10
}
```

You may encounter the following errors: *(list of errors would be here)*

Power Off a Server
To power off a server, use the following request:

HTTP Request
```http
POST /ecs/v2/servers/{servername}/actions
{
  "type": "power_off"
}
```

HTTP Response
```json
{
```

```
  "id": 709,
  "status": "in-progress",
  "user": "foo",
  "created_at": "2024-10-19T15:08:55.887799+00:00",
  "started_at": "",
  "completed_at": "",
  "resource": "ec200001",
  "resource_type": "ECS",
  "type": "power_off",
  "progress": 10
}
```

You may encounter the following errors: *(list of errors would be here)*

Restore a Snapshot
To restore a snapshot, simply execute the following request:

HTTP Request
```http
{
  "type": "rollback",
  "snapshot": 105
}
```

HTTP Response
```json
{
  "id": 2118,
  "status": "in-progress",
  "user": "admin",
  "created_at": "2025-02-07T13:46:27.864204+00:00",
  "started_at": "2025-02-07T13:46:27.863404+00:00",
  "completed_at": null,
  "resource": "ec200409",
  "resource_type": "ECS",
  "type": "rollback",
  "progress": 0
}
```

Note
This operation will require a server reboot and may take several minutes.

DOC: hosting/cloudserver/rest-api/API-Endpoints/Networks/index.txt

Networks
Networks are essential tools that allow you to connect your virtual servers through a secure, private, and isolated channel in a simple and abstracted way.

Create a Network
To connect your servers to a network, you first need to create one. Use the following request:

HTTP Request
```http
POST /ecs/v2/networks
```

HTTP Response
```json
{
  "status": "created",
  "network": {
    "id": 37,
    "network_id": 17,
    "name": "net000017", // identifier name of the network
    "active": true,
    "created_at": "2024-11-19T08:12:13.884Z",
    "deactivated_at": null,
    "user": "admin" // user to whom the network belongs
  }
}
```

List Networks
You can list all active networks with the request:

HTTP Request
```http
GET /ecs/v2/networks
```

HTTP Response
```json
{
  "status": "ok",
  "networks": [
    {
      "id": 35,
      "network_id": 15,
      "name": "net000015",
      "active": true,
      "created_at": "2024-11-15T16:03:45.822Z",
      "deactivated_at": null,
      "user": "admin"
    },
    {
      "id": 36,
      "network_id": 16,
      "name": "net000016",
      "active": true,
      "created_at": "2024-11-18T09:05:59.322Z",
      "deactivated_at": null,
```

```
      "user": "admin"
    }
  ]
}
```

View a Specific Network
Alternatively, to view a specific network you can use the call:

HTTP Request
```http
GET /ecs/v2/networks/{name}
```

HTTP Response
(response example would be here)
```
{
"status": "ok",
"network": {
"id": 37,
"network_id": 17,
"name": "net000017",
"active": true,
"created_at": "2024-11-19T08:12:13.884Z",
"deactivated_at": null,
"user": "admin"
}
}
```
Remove a network
The ability to remove a network is enabled only if there are no active connections to it. To remove a script you can use the following API call:
HTTP Request
DELETE /ecs/v2/networks/net000017
HTTP Response
```
{
"status": "no_content"
}
```
You may receive the following errors
vlans
To connect a server to a network you need to generate a VLAN between the server and the desired network.
Creating the first “connection”, i.e., the first VLAN between a server and a network, will cause the server to reboot.
A reboot will also be required to remove any VLAN present between a network and a server.
Create a VLAN
To use a VLAN you must create it using the following request:
HTTP Request
POST /ecs/v2/servers/{servername}/networks/{networkname}/vlans
```
{
"vlan_id": 1, //l'id della vlan
"pvid": true //se la vlan è pvid dell'interfaccia su cui è ospite
}
```
HTTP Response

```
{
"status": "creating",
"action": {
"id": 1552,
"status": "in-progress",
"user": "admin",
"created_at": "2024-11-20T09:59:46.353012+00:00",
"started_at": null,
"completed_at": null,
"resource": "ec200322-net000022",
"resource_type": "network",
"type": "add_vlan",
"progress": 0
}
}
```


You may receive the following errors:

Info

The configuration shown as an example is the most classic and is generated via the panel when adding a new interface.

Create a VLAN range

If you want to quickly create a range of VLANs, the following request is available:

HTTP Request


```
POST /ecs/v2/servers/{servername}/networks/{networkname}/vlans
{
"vlans": "1-100",
}
```


HTTP Response


```
{
"status": "creating",
"action": {
"id": 1552,
"status": "in-progress",
"user": "admin",
"created_at": "2024-11-20T09:59:46.353012+00:00",
"started_at": null,
"completed_at": null,
"resource": "ec200322-net000022",
"resource_type": "network",
"type": "add_vlan",
"progress": 0
}
}
```


You may receive the following errors:

Get all VLANs of a server on a network

After creating VLANs you can view them with the following request:

HTTP Request


```
GET /ecs/v2/servers/{servername}/networks/{networkname}/vlans
```


HTTP Response


```
{
"status": "ok",
"vlans": [
{
"id": 132021,
```

```
"vlan_id": 1,
"pvid": true,
"status": "ok",
"active": true,
"created_at": "2024-11-20T09:59:46.347Z",
"deactivated_at": null,
"mac_address": "52:54:00:14:f9:1e",
"network": "net000022",
"server": "ec200322"
},
{
"id": 132022,
"vlan_id": 2,
"pvid": false,
"status": "ok",
"active": true,
"created_at": "2024-11-20T10:17:52.338Z",
"deactivated_at": null,
"mac_address": "52:54:00:14:f9:1e",
"network": "net000022",
"server": "ec200322"
}
]
}
```


Update a VLAN

To update a VLAN and define a new PVID, the following request is available:

HTTP Request


```
PATCH /ecs/v2/servers/{servername}/networks/{networkname}/vlans/{vlan_id}
{
"pvid": true
}
```


HTTP Response


```
{
"status": "updating",
"vlan": {
"id": 132022,
"server": "ec200322",
"network": "net000022",
"vlan_id": 2,
"status": "updating",
"active": true,
"created_at": "2024-11-20T10:17:52.338Z",
"deactivated_at": null,
"mac_address": "52:54:00:14:f9:1e",
"pvid": true,
"user": "admin"
},
"action": {
"id": 1554,
"status": "in-progress",
"user": "admin",
"created_at": "2024-11-20T10:24:57.106901+00:00",
"started_at": null,
```

```
"completed_at": null,
"resource": "ec200322-net000022",
"resource_type": "network",
"type": "update_vlan",
"progress": 0
}
}
```

Warning

A VLAN must have a status of “ok” to be updated or removed

Remove a VLAN

Obviously you can also remove a VLAN; to do so the following request is available:

HTTP Request

```
DELETE
/ecs/v2/servers/{servername}/networks/{networkname}/vlans/{vlan_id}
```

Remove a range of VLANs

Similarly you can remove a range of VLANs with the following call:

HTTP Request

```
DELETE
/ecs/v2/servers/{servername}/networks/{networkname}/vlans/{vlan_id_start}
-{vlan_id_end}
```

You may receive the following errors:

```
================================================================================
========
```

```
DOC: hosting/cloudserver/rest-api/API-Endpoints/Plan/index.txt
----------------------------------------------------------------
```

Plans

Plans define classes of resources on which to build a server; several types exist:

- Shared CPU
- High Memory
- GPU
- NPU

Each type represents a different product but all can be activated via the ECS API or the Ermes web panel.

Get all Plans

To retrieve all plans available to your user, simply use the following request:

HTTP Request

```
GET /ecs/v2/plans
```

HTTP Response

```
"status": "ok",
"plans": [
{
"id": 1, //plan id
"name": "ECS1", //plan name
"cpu": "1",
"ram": "1024",
"disk": "20",
"gpu": "0",
"gpu_label": null,
```

```
"hourly_price": 0.017, //price for hour
"montly_price": 12.0,
"windows": false, //is a windows plan
"host_type": "ECS", //required host type
"available": true, //plan active
"available_regions": [ //all region where a plan is active
{
"id": 1,
"location": "it-fr2",
"description": "Frosinone"
}
]
},
{
"id": 2,
"name": "ECS2",
"cpu": "2",
"ram": "2048",
"disk": "40",
"gpu": "0",
"gpu_label": null,
"hourly_price": 0.028,
"montly_price": 20.0,
"windows": false,
"host_type": "ECS",
"available": true,
"available_regions": [
{
"id": 1,
"location": "it-fr2",
"description": "Frosinone"
}
]
},
]
}
```

Warning

The presence of a region in the list of available regions does not necessarily mean that the plan can actually be activated in that location; it only indicates that the region is compatible with the specific plan.

Actually activatable plans

To know which plans are truly activatable and avoid possible resource-availability errors, you can use the following request:

HTTP Request

GET /ecs/v2/plans/availables

HTTP Response

```json
{
  "status": "ok",
  "plans": [
    {
      "id": 1,
      "name": "ECS1",
```

```
      "cpu": "1",
      "ram": "1024",
      "disk": "20",
      "gpu": "0",
      "gpu_label": null,
      "hourly_price": 0.017,
      "montly_price": 12.0,
      "windows": false,
      "host_type": "ECS",
      "available": true,
      "os_availables": [ // which base images can be used to activate the
plan
        {
          "id": 1,
          "name": "centos-7",
          "creation_date": "2019-04-16T13:37:41+00:00",
          "active_flag": true,
          "status": "CD",
          "uuid": "ffe2a034-7a44-4f51-9ae8-17fb354793f6",
          "description": "CentOS Linux 7",
          "notes": "Linux CentOS disk image",
          "public": true,
          "cloud_image": false,
          "so_base": "centos",
          "api_version": "v4",
          "version": "7"
        },
        {
          "id": 2,
          "name": "debian-9",
          "creation_date": "2019-04-18T12:34:08+00:00",
          "active_flag": true,
          "status": "CD",
          "uuid": "6708cdc4-0c8d-4ccf-83c7-ae39bccaeb42",
          "description": "Debian Linux 9",
          "notes": "Linux Debian disk image",
          "public": true,
          "cloud_image": false,
          "so_base": "debian",
          "api_version": "v4",
          "version": "9"
        }
      ],
      "region_availables": [ // in which regions the plan can currently
be activated
        {
          "region": "it-fr2",
          "hosts": [ // tells on which physical host the machine can be
activated
            {
              "host": "ecs911.swlab.seeweb.it",
              "servers": []
            },
            {
```

```
              "host": "ecs912.swlab.seeweb.it",
              "servers": [ // servers to know from which servers the new
server can be isolated
                "ec000025"
              ]
            }
          ]
        }
      ]
    }
    ...
  ]
}
```

Info
For most activations the `host` field may be irrelevant, but if you want to use isolation it becomes extremely important.

In the example above, if I wanted to isolate the new server from `ec000025` I know I can do it because, besides that server, there is another host available for my plan. If there had been only one host, isolation would not have been possible; the request would have failed with a resource-shortage error.

Remember: by using isolation you are explicitly telling the system **not** to look for free resources on physical hosts that belong to other virtual servers you own. Therefore, verify that isolation is actually feasible before using it via the API.

DOC: hosting/cloudserver/rest-api/API-Endpoints/Regions/index.txt

Regions
Regions, also referred to as datacenters, play a fundamental role in our services because they determine the geographic location of the servers you will create. If it is not possible to create a server in a given region, changing the datacenter may solve the resource-availability problem.

All Regions
To discover which regions are available, use the following request:

HTTP Request
```
GET /ecs/v2/regions
```

HTTP Response
```json
{
  "status": "ok",
  "regions": [
```

```
    {
      "location": "ch-lug1",   // region code
      "description": "Lugano" // region label
    },
    {
      "location": "hr-zag1",
      "description": "Zagreb"
    },
    {
      "location": "it-fr2",
      "description": "Frosinone"
    },
    {
      "location": "it-mi2",
      "description": "Milan"
    }
  ]
}
```

Available Regions for a Specific Plan
Obviously, certain plans may not be available in all regions. To check the availability of a plan in a specific region, use the following request:

HTTP Request
```
POST /ecs/v2/regions/availables
{
  "plan":"ECS6"
}
```

HTTP Response
```json
{
  "status": "ok",
  "regions": [ // regions where that plan is available
    "it-fr2"
  ]
}
```

Warning
If you try to create a plan in a region that is not available, the request will be accepted but the resulting action will fail due to lack of resources.

DOC: hosting/cloudserver/rest-api/API-Endpoints/SSH-Keys/index.txt

SSH Keys

Even though One-Time Passwords are used as the basic authentication method for a new server, we strongly recommend using SSH keys for authentication.

Our services expose several endpoints that allow you to manage your library of public SSH keys.

Adding an SSH Key
To add an SSH key, first generate it locally. Once the key is generated, add it to your user with the following request:

HTTP Request
```
POST /ecs/v2/sshkeys
{
  "key": "ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQDk5S4uuptEa3ylKjIVZIanNu+FYayXETb4a5R+0YfGW
0ifTjDObH0bKFPi0/Val8NzbN6eSEPA0h5pH/jUlnxLIhcXd/bqZyp4NXlP8k2ls+O7Z6m24S
4O6/lOrMnFxiDkx6S0YtVCkAzFgcH8mUDYWMmFQd3YFF6e8KIyB+IZKJc/vfd6QiV/2Zo8fF0
z+69YDnjvsiUCvegTjg90PatG2q7WmlRZbxbcb5CJYRqi2cNUX7FejwTBvxH4G82cRf8ozyQn
u60Ir2eOL0GqfEmObbbC7om8lMwqptVN6SlYfXtJovqX36xFXvUX+U+lLXv/LeysaOnGRLKp7
5x+oyCOXPmG7XkZPTGcaS/k/oZfWYvMwQwuBZhH+bFk3lZ5xWtSJqNXh5H+zemBW/Eu71JN4S
e2Hm3bjBL2dR4HE4eef282Sk1jW3U3bdTaADF96QQaoF630r70HeaqhuUiIdM//1guqF5NhIK
9AVu3pXVdKnW3XEEuEzr1RUVBrJUcgTk= user@my-pc",
  "label": "my-ssh-key"
}
```

HTTP Response
```json
{
  "status": "ok"
}
```

You may encounter the following errors:

Warning
Unlike the Ermes panel, key validation is **not** performed via the API; it is the responsibility of the user who submits the key. An invalid key will not work and could compromise the server configuration.

Retrieving All Active SSH Keys
You can obtain a list of all active SSH keys with the following request:

HTTP Request
```
GET /ecs/v2/sshkeys
```

HTTP Response
```json
{
  "status": "ok",
```

```
  "pubkeys": [
    {
      "id": 0,
      "label": "my-pub-key-01",
```
```json
{
  "key": "ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQDk5S4uuptEa3ylKjIVZIanNu+FYayXETb4a5R+0YfGW
0ifTjDObH0bKFPi0/Val8NzbN6eSEPA0h5pH/jUlnxLIhcXd/bqZyp4NXlP8k2ls+O7Z6m24S
4O6/lOrMnFxiDkx6S0YtVCkAzFgcH8mUDYWMmFQd3YFF6e8KIyB+IZKJc/vfd6QiV/2Zo8fF0
z+69YDnjvsiUCvegTjg90PatG2q7WmlRZbxbcb5CJYRqi2cNUX7FejwTBvxH4G82cRf8ozyQn
u60Ir2eOL0GqfEmObbbC7om8lMwqptVN6SlYfXtJovqX36xFXvUX+U+lLXv/LeysaOnGRLKp7
5x+oyCOXPmG7XkZPTGcaS/k/oZfWYvMwQwuBZhH+bFk3lZ5xWtSJqNXh5H+zemBW/Eu71JN4S
e2Hm3bjBL2dR4HE4eef282Sk1jW3U3bdTaADF96QQaoF630r70HeaqhuUiIdM//1guqF5NhIK
9AVu3pXVdKnW3XEEuEzr1RUVBrJUcgTk= user@my-pc-1",
  "created_at": "2024-06-28T07:59:56.810000+00:00"
},
{
  "id": 1,
  "label": "my-pub-key-02",
  "key": "ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQDk5S4uuptEa3ylKjIVZIanNu+FYayXETb4a5R+0YfGW
0ifTjDObH0bKFPi0/Val8NzbN6eSEPA0h5pH/jUlnxLIhcXd/bqZyp4NXlP8k2ls+O7Z6m24S
4O6/lOrMnFxiDkx6S0YtVCkAzFgcH8mUDYWMmFQd3YFF6e8KIyB+IZKJc/vfd6QiV/2Zo8fF0
z+69YDnjvsiUCvegTjg90PatG2q7WmlRZbxbcb5CJYRqi2cNUX7FejwTBvxH4G82cRf8ozyQn
u60Ir2eOL0GqfEmObbbC7om8lMwqptVN6SlYfXtJovqX36xFXvUX+U+lLXv/LeysaOnGRLKp7
5x+oyCOXPmG7XkZPTGcaS/k/oZfWYvMwQwuBZhH+bFk3lZ5xWtSJqNXh5H+zemBW/Eu71JN4S
e2Hm3bjBL2dR4HE4eef282Sk1jW3U3bdTaADF96QQaoF630r70HeaqhuUiIdM//1guqF5NhIK
9AVu3pXVdKnW3XEEuEzr1RUVBrJUcgTk= user@my-pc-2",
  "created_at": "2024-06-28T09:59:56.810000+00:00"
}
]
}
```

Get a specific SSH key
You can also retrieve a specific key by using its ID.

HTTP Request
`GET /ecs/v2/sshkeys/{keyid}`

HTTP Response
```json
{
  "status": "ok",
  "pubkey": {
    "id": 0,
    "label": "my-pub-key-01",
    "key": "ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQDk5S4uuptEa3ylKjIVZIanNu+FYayXETb4a5R+0YfGW
0ifTjDObH0bKFPi0/Val8NzbN6eSEPA0h5pH/jUlnxLIhcXd/bqZyp4NXlP8k2ls+O7Z6m24S
4O6/lOrMnFxiDkx6S0YtVCkAzFgcH8mUDYWMmFQd3YFF6e8KIyB+IZKJc/vfd6QiV/2Zo8fF0
z+69YDnjvsiUCvegTjg90PatG2q7WmlRZbxbcb5CJYRqi2cNUX7FejwTBvxH4G82cRf8ozyQn
```

```
u60Ir2eOL0GqfEmObbbC7om8lMwqptVN6SlYfXtJovqX36xFXvUX+U+lLXv/LeysaOnGRLKp7
5x+oyCOXPmG7XkZPTGcaS/k/oZfWYvMwQwuBZhH+bFk3lZ5xWtSJqNXh5H+zemBW/Eu71JN4S
e2Hm3bjBL2dR4HE4eef282Sk1jW3U3bdTaADF96QQaoF630r70HeaqhuUiIdM//1guqF5NhIK
9AVu3pXVdKnW3XEEuEzr1RUVBrJUcgTk= user@my-pc-1",
    "created_at": "2024-06-28T07:59:56.810000+00:00"
  }
}
```

Rename an SSH key
You can change the `label` field of an SSH key with the following request:

HTTP Request
`PUT /ecs/v2/sshkeys/{keyid}`
```json
{
  "label": "my-renamed-key"
}
```

HTTP Response
```json
{
  "status": "ok",
  "pubkey": {
    "id": 0,
    "label": "my-renamed-key",
    "key": "ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQDk5S4uuptEa3ylKjIVZIanNu+FYayXETb4a5R+0YfGW
0ifTjDObH0bKFPi0/Val8NzbN6eSEPA0h5pH/jUlnxLIhcXd/bqZyp4NXlP8k2ls+O7Z6m24S
4O6/lOrMnFxiDkx6S0YtVCkAzFgcH8mUDYWMmFQd3YFF6e8KIyB+IZKJc/vfd6QiV/2Zo8fF0
z+69YDnjvsiUCvegTjg90PatG2q7WmlRZbxbcb5CJYRqi2cNUX7FejwTBvxH4G82cRf8ozyQn
u60Ir2eOL0GqfEmObbbC7om8lMwqptVN6SlYfXtJovqX36xFXvUX+U+lLXv/LeysaOnGRLKp7
5x+oyCOXPmG7XkZPTGcaS/k/oZfWYvMwQwuBZhH+bFk3lZ5xWtSJqNXh5H+zemBW/Eu71JN4S
e2Hm3bjBL2dR4HE4eef282Sk1jW3U3bdTaADF96QQaoF630r70HeaqhuUiIdM//1guqF5NhIK
9AVu3pXVdKnW3XEEuEzr1RUVBrJUcgTk= user@my-pc-1",
    "created_at": "2024-06-28T07:59:56.810000+00:00"
  }
}
```

Delete an SSH key
To delete an SSH key, use the following request:

HTTP Request
`DELETE /ecs/v2/sshkeys/{keyid}`

HTTP Response
```json
{
  "status": "ok"
}
```

DOC: hosting/cloudserver/rest-api/API-Endpoints/Servers/index.txt
Server
Virtual servers are the core of the ECS services. This page lists all the operations available on servers.

Create a new server
Very little information is required to create a basic server; you only need:
- A Plan
- An Image or a Template
- A Region or Location
- A Label

HTTP Request
`POST /ecs/v2/servers`
```json
{
  "plan": "ECS6",
  "image": "debian-12",
  "location": "it-fr2",
  "notes": "my-first-server"
}
```

HTTP Response
```json
{
  "status": "ok",
  "action": 35,
  "server": {
    "name": "ec200001",
    "ipv4": "",
    "ipv6": "",
    "plan": "ECS6",
    "location": "it-fr2",
    "os": "debian-12",
    "status": "Booting",
    "progress": 10
    ...
  }
}
```

You can monitor the creation Action status to know when a server is ready or if the operation failed.

Note
Server creation can easily fail due to lack of resources. To avoid this, it is recommended to check Plan availability before attempting to create a server in a specific data center.

List all servers
To list all your servers you can simply use the following endpoint:

HTTP Request
`GET /ecs/v2/servers`

HTTP Response
```json
{
  "status": "ok",
  "count": 5,
  "server": [
    {
      "name": "ec200001",
      "ipv4": "127.0.0.1",
      "ipv6": "fe80::1",
      "group": "eg100001",
      "plan": "ECS6",
      "plan_size": {
        "core": 12,
        "ram": 24576,
        "disk": 400,
        "gpu": 0,
        "gpu_label": "",
        "host_type": "ECS"
      },
      "reserved_plans": [],
      "is_reserved": false,
      "reserved_until": "",
      "support": null,
      "location": "it-fr2",
      "location_label": "frosinone",
      "notes": "my-server",
      "so": "debian-12",
      "so_label": "Debian 12",
      "creation_date": "2024-10-10 12:00...",
      "deletion_date": null,
      "active_flag": true,
      "status": "Booted",
      "progress": 100,
      "api_version": "v4",
      "user": "eCS99999"
    },
    ...
  ]
}
```

Get a specific server
To retrieve a single server, add the server name to the URL as follows:

HTTP Request
`GET /ecs/v2/servers/{servername}`

HTTP Response
```json
{
  "status": "ok",
  "server": {
    "name": "ec200001",
    "ipv4": "127.0.0.1",
    "ipv6": "fe80::1",
    "group": "eg100001",
    "plan": "ECS6",
    "plan_size": {
      "core": 12,
      "ram": 24576,
      "disk": 400,
      "gpu": 0,
      "gpu_label": "",
      "host_type": "ECS"
    },
    "reserved_plans": [],
    "is_reserved": false,
    "reserved_until": "",
    "support": null,
    "location": "it-fr2",
    "location_label": "frosinone",
    "notes": "my-server",
    "so": "debian-12",
    "so_label": "Debian 12",
    "creation_date": "2024-10-10 12:00...",
    "deletion_date": null,
    "active_flag": true,
    "status": "Booted",
    "progress": 100,
    "api_version": "v4",
    ...
  }
}
```
```json
"user": "eCS99999",
}
}
```

Change Server Label
If you want to change a label for a specific server, you can use the following request:

HTTP Request
`PUT /ecs/v2/servers/{servername}`
```json
{
  "note": "my-new-server-label"
}
```

HTTP Response
```json
{
  "status": "ok"
}
```

Reserved Period
Subscribing to a Reserved Period is a good choice if you face challenges that require high work intensity for medium-to-long periods. By doing so, you can receive significant discounts based on the duration of the period, disabling the ability to destroy the server.

Add a Reserved Period During Creation
To subscribe to a Reserved Period when creating a server, add the `reserved_plan` field to your request as follows:

HTTP Request
`POST /ecs/v2/servers`
```json
{
  "plan": "ECS6",
  "image": "debian-12",
  "location": "it-fr2",
  "notes": "my-first-server",
  "reserved_plan": "M24PECS6"
}
```

The `reserved_plan` field is composed of:
`M{plan duration}P{Plan name}`
where the plan duration can be **3**, **6**, or **12** months.

Add a Reserved Period to an Existing Server
To add a Reserved Period to an existing server, use the following request:

HTTP Request
`POST /api/v2/servers/{servername}/reserved`
```json
{
  "reserved_plan": "M24PECS6"
}
```

HTTP Response
```json
{
  "reserved_plan": 50,
  "reserved_month": 24,
  "plan": 7,
  "discount": 20,
  "start_date": "2024-10-10 12:34...",
```

```
  "end_date": "2026-10-10 12:34...",
  "server": "ec200001"
}
```

You may encounter the following errors:

Support
Choosing a support plan for your server allows you to receive help directly from our expert engineers; several support plans are available. For more information, consult our website.

Add a Support Plan During Creation
To add a Support Plan when creating a server, include the `support` field in your request:

HTTP Request
`POST /ecs/v2/servers`
```json
{
  "plan": "ECS6",
  "image": "debian-12",
  "location": "it-fr2",
  "notes": "my-first-server",
  "support": "global"
}
```

The `support` field can be: `"basic"` or `"global"`.

To activate “proactive” support, we recommend visiting our website and contacting our offices.

Add or Modify a Support Plan on an Existing Server
To add or change a Support Plan on an existing server, use the following request:

HTTP Request
`POST /ecs/v2/servers/{server-name}/support`
```json
{
  "code": "global"
}
```

HTTP Response
```json
{
  "status": "ok",
  "support": {
    "support_title": "global",
    "support_code": "global",
```

```
    "immutable": false, // if forced not to change
    "start": "2024-10-10 17:03...",
    "start": null,
    "may_downgrade": false, // if the plan can be downgraded
    "weigth": 2, // the weight of the plan is used to determine
downgrade/upgrade relationships
    "days": 30, // duration
    "cancelled": false,
    "cancelled_at": null
  }
}
```

You can also use this endpoint to update or downgrade your support plan.

You may encounter the following errors:

Remove a Support Plan
To remove a support plan (and set the support to unmanaged), simply use the following request:

HTTP Request
`DELETE /ecs/v2/servers/{server-name}/support`

HTTP Response
```json
{
  "status": "ok"
}
```

You may encounter the following errors:

Groups
Groups allow you to have a segmented view of your servers via the Ermes web panel; they can also be configured through the following API endpoints:

Info
For more information on creating groups, see the **Group Page**.

Add a Server to a Group During Creation
To add your server to a group when creating it, use the `group` field:

HTTP Request
`POST /ecs/v2/servers`
```json
{
  "plan": "ECS6",
  "image": "debian-12",
  "location": "it-fr2",
  "notes": "my-first-server",
```

```
  "group": "eg-000001"
}
```

Add an Existing Server to a Group
To add an existing server to a group, use the server-management endpoint:

HTTP Request
`PUT /ecs/v2/servers/{servername}`
```json
{
  "group": "eg-000001"
}
```

HTTP Response
```json
{
  "status": "ok"
}
```

You may encounter the following errors:

Remove a Server from a Group
To remove a server from a group, use the following request:

HTTP Request
`PUT /ecs/v2/servers/{servername}`
```json
{
  "group": "nogroup"
}
```

HTTP Response
```json
{
  "status": "ok"
}
```

Cloud Script
Cloud Script is a powerful feature that lets you run a remote script during server installation. For more information on Cloud Script, visit the **Cloud Script Page**.

You can use a Cloud Script when creating a server either as a **Temporary Script** or as an existing script from your library. Using the script library is strongly recommended, especially when using the API.

Temporary Script

Using a Temporary (“otf”) Script means sending the entire script content in the server-creation request:

HTTP Request
`POST /ecs/v2/servers`
```json
{
  "plan": "ECS6",
  "image": "debian-12",
  "location": "it-fr2",
  "notes": "my-first-server",
  "user_customize": "#!/bin/bash\necho \"my script\" > /root/out.txt"
}
```

Script Library and Environment Variables
Using the Script Library requires that a Cloud Script is already written in your Script Library. It can then be referenced during creation.

For example, consider the following script with ID **2**:
```bash
#!/bin/bash
echo "i'm $NAME, i work in $WORK !!!" > /root/presentation.txt
```

You can use it via:

HTTP Request
`POST /ecs/v2/servers`
```json
{
  "plan": "ECS6",
  "image": "debian-12",
  "location": "it-fr2",
  "notes": "my-first-server",
  "user_customize": 2, // script code
  "user_customize_env": "NAME=\"bob\"\nWORK=\"seeweb\""
}
```

The environment variables (`user_customize_env`) allow you to supply the script’s content and modify only the necessary values; they are treated as sensitive information and are masked accordingly.
The environment variables must follow this syntax to work:
`VARNAME1=VARCONTENT1\nVARNAME2=VARCONTENT2...`.
You may receive the following errors:

Use SSH Keys
To use SSH-key authentication via the API, you must first register your public SSH key and use it through the following field when creating the server:

HTTP Request
```http
```

```
POST /ecs/v2/servers
{
  "plan": "ECS6",
  "image": "debian-12",
  "location": "it-fr2",
  "notes": "my-first-server",
  "ssh_key": "my-key-label"
}
```

Add a Network during creation
Networks are an extremely useful feature that introduces networking concepts to ECS products. To create and manage networks, see *Networks*. To add networks during creation, simply use the `networks` field when creating the server:

HTTP Request
```http
POST /ecs/v2/servers
{
  "plan": "ECS6",
  "image": "debian-12",
  "location": "it-fr2",
  "notes": "my-first-server",
  "networks": [
    {
      "name": "net000015", // network name
      "vlans": [ // list of VLANs
        {
          "vlan_id": 100, // single VLAN
          "pvid": true // if the VLAN is PVID
        },
        {
          "vlans": "1-99" // VLAN range
        },
        {
          "vlans": "150-200"
        }
      ]
    }
  ]
}
```

Isolation
Isolation is a very interesting feature that allows you to separate physical hardware resources between two servers in the same datacenter. This way you can guarantee the security of multiple instances at the physical-resource level.

Set Isolation
You can isolate a server by adding the following field during its creation:

HTTP Request
```http
POST /ecs/v2/servers
{
  "plan": "ECS6",
  "image": "debian-12",
  "location": "it-fr2",
  "notes": "my-isolated-server",
  "isolate_from": [
    "ecs000001",
    "ecs000002"
  ]
}
```

This forces the new server to start on a physical host different from `ecs000001` and `ecs000002`.

Get all isolations of a server
To retrieve all active isolations for a server, you can use the following request:

HTTP Request
```
/ecs/v2/servers/{servername}/isolations
```

HTTP Response
```json
{
  "status": "ok",
  "isolations": [
    {
      "name": "ec000001",
      "notes": "my-isolated-server"
    },
    {
      "name": "ec000002",
      "notes": "my-fabulous-isolated-server"
    }
  ]
}
```

Get server status
To obtain the operational status of the server, you can use this endpoint:

HTTP Request
```
GET /ecs/v2/servers/{servername}/status
```

HTTP Response

```json
{
  "status": "ok",
  "server": {
    "name": "ec000001",
    "current_status": "Running"
  }
}
```

You may receive the following errors:

Metrics
The ECS service exports various performance information and metrics for your server. You can obtain CPU, Disk, and Network usage via the following endpoint:

HTTP Request
```
GET /ecs/v2/servers/{servername}/metrics
```

HTTP Response
```json
{
  "metrics": {
    "resource": "ec000001",
    "core": [
      {
        "t": "2024-10-17 08:25:58",
        "y": "0.00"
      },
      {
        "t": "2024-10-17 08:30:58",
        "y": "0.00"
      }
      ...
    ],
    "disk": {
      "reads": [
        {
          "t": "2024-10-17 08:35:58",
          "y": "28.45"
        },
        {
          "t": "2024-10-17 08:40:58",
          "y": "0.00"
        }
        ...
      ],
      "writes": [
        {
          "t": "2024-10-17 08:35:58",
          "y": "2.37"
```

```
        },
        {
          "t": "2024-10-17 08:40:58",
          "y": "0.00"
        }
        ...
      ]
    },
    "network": {
      "received": [
        {
          "t": "2024-10-17 08:35:58",
          "y": "0.00"
        },
        {
          "t": "2024-10-17 08:40:58",
          "y": "0.00"
        }
        ...
      ],
      "transmitted": [
        {
          "t": "2024-10-17 08:35:58",
          "y": "0.00"
        },
        {
          "t": "2024-10-17 08:40:58",
          "y": "0.00"
        }
        ...
      ]
    }
  }
}
```

Destroy a Server
To destroy a server, the following request is available:

HTTP Request
```
DELETE /ecs/v2/servers/{servername}
```

HTTP Response
```json
{
  "status": "ok",
  "action": {
    "id": 39,
    "status": "in-progress",
    "user": "foo",
    "created_at": "2019-04-30T16:33:03.317800+00:00",
    "started_at": "2019-04-30T16:33:03.317019+00:00",
```

```
    "completed_at": null,
    "resource": "ec200016",
    "resource_type": "ECS",
    "type": "delete_server",
    "progress": 0
  }
}
```

DOC: hosting/cloudserver/rest-api/API-Endpoints/Snapshots/index.txt

Snapshot
Snapshots are simply copies of your virtual server. Like Templates, they retain all data of your machine but have the following differences:

- You can restore a server's state to a Snapshot.
- A Snapshot cannot be used to create a new machine.
- A Snapshot can be converted into a Template, but not vice-versa.

Protected Snapshots
Snapshots are divided into protected and unprotected snapshots and have the following characteristics:

Managing Snapshots
Our services provide several HTTP calls to manage Snapshots. Below are the most common operations:

Create a Snapshot
To create a snapshot, the following request is available:

HTTP Request
```http
POST /ecs/v2/servers/{servername}/snapshots
{
  "description": "my-first-snapshot",
  "notes": "my-first-snapshot"
}
```

HTTP Response
```json
{
  "status": "ok",
  "snapshot": {
    "id": 104,
    "name": "ec200409-SNP-1",
    "user": "admin",
    "snapshot_parent": null,
    "snapshot_parent_name": null,
    "is_last_restored": false,
    "protected": true,
    "restoring": false,
```

```
    "source_server": "ec200409",
    "status": "CG",
    "status_label": "Creating",
    "uid": "eb7e9539-3f77-4ebc-811d-58237521a8cc",
    "description": "my-first-snapshot",
    "notes": "my-first-snapshot",
    "active_flag": false,
    "size_on_disk": null,
    "created_at": "2025-02-07T12:50:02.840Z",
    "updated_at": "2025-02-07T12:50:02.840Z",
    "deleted_at": null,
    "api_version_value": 5,
    "api_version": "v5"
  },
  "action": {
    "id": 2115,
    "status": "in-progress",
    "user": "admin",
    "created_at": "2025-02-07T12:50:02.846870+00:00",
    "started_at": "2025-02-07T12:50:02.846406+00:00",
    "completed_at": null,
    "resource": "104",
    "resource_type": "SNP",
    "type": "create_snapshot",
    "progress": 0
  }
}
```

List all Snapshots
To list all Snapshots, the following call is available:

HTTP Request
```
GET /ecs/v2/servers/{servername}/snapshots
```

HTTP Response
```json
{
  "status": "ok",
  "snapshots": [
    {
      "id": 104,
      "name": "ec200409-SNP-1",
      ...
    }
  ]
}
```
```json
"user": "admin",
"snapshot_parent": null,
"snapshot_parent_name": null,
```

```
"is_last_restored": true,
"protected": true,
"restoring": false,
"source_server": "ec200409",
"status": "CD",
"status_label": "Created",
"uid": "eb7e9539-3f77-4ebc-811d-58237521a8cc",
"description": "my-first-snapshot",
"notes": "my-first-snapshot",
"active_flag": true,
"size_on_disk": 3804560,
"created_at": "2025-02-07T12:50:02.840Z",
"updated_at": "2025-02-07T12:52:03.301Z",
"deleted_at": null,
"api_version_value": 5,
"api_version": "v5"
}
]
}
```

Get a Specific Snapshot
To retrieve a specific snapshot, use the following request:

HTTP Request

```
GET /ecs/v2/servers/{servername}/snapshots/{snapshot_id}
```

HTTP Response

```json
{
  "status": "ok",
  "snapshot": {
    "id": 104,
    "name": "ec200409-SNP-1",
    "user": "admin",
    "snapshot_parent": null,
    "snapshot_parent_name": null,
    "is_last_restored": true,
    "protected": true,
    "restoring": false,
    "source_server": "ec200409",
    "status": "CD",
    "status_label": "Created",
    "uid": "eb7e9539-3f77-4ebc-811d-58237521a8cc",
    "description": "my-first-snapshot",
    "notes": "my-first-snapshot",
    "active_flag": true,
    "size_on_disk": 3804560,
    "created_at": "2025-02-07T12:50:02.840Z",
    "updated_at": "2025-02-07T12:52:03.301Z",
```

```
    "deleted_at": null,
    "api_version_value": 5,
    "api_version": "v5"
  }
}
```

Modify a Snapshot
You can also update some information of a snapshot:

HTTP Request

```
PATCH /ecs/v2/servers/{servername}/snapshots/{snapshot_id}
{
  "description": "new-snap-description", // update snapshot description
  "notes": "new-snap-notes",               // update snapshot notes
  "protected": true                        // protect the snapshot
}
```

HTTP Response

```json
{
  "status": "ok",
  "snapshot": {
    "id": 104,
    "name": "ec200409-SNP-1",
    "user": "admin",
    "snapshot_parent": null,
    "snapshot_parent_name": null,
    "is_last_restored": true,
    "protected": true,
    "restoring": false,
    "source_server": "ec200409",
    "status": "CD",
    "status_label": "Created",
    "uid": "eb7e9539-3f77-4ebc-811d-58237521a8cc",
    "description": "new-snap-description",
    "notes": "new-snap-description",
    "active_flag": true,
    "size_on_disk": 3804560,
    "created_at": "2025-02-07T12:50:02.840Z",
    "updated_at": "2025-02-07T13:11:20.196Z",
    "deleted_at": null,
    "api_version_value": 5,
    "api_version": "v5"
  }
}
```

Delete a Snapshot
When a snapshot is no longer needed, you can destroy it:

HTTP Request

```
DELETE /ecs/v2/servers/ec200409/snapshots/104
```

HTTP Response

```json
{
  "status": "ok",
  "snapshot": {
    "id": 104,
    "name": "ec200409-SNP-1",
    "user": "admin",
    "snapshot_parent": null,
    "snapshot_parent_name": null,
    "is_last_restored": true,
    "protected": true,
    "restoring": false,
    "source_server": "ec200409",
    "status": "DE",
    "status_label": "Deleting",
    "uid": "eb7e9539-3f77-4ebc-811d-58237521a8cc",
    "description": "new-snap-description",
    "notes": "new-snap-description",
    "active_flag": false,
    "size_on_disk": 3804560,
    "created_at": "2025-02-07T12:50:02.840Z",
    "updated_at": "2025-02-07T13:12:44.450Z",
    "deleted_at": "2025-02-07T13:12:44.447Z",
    "api_version_value": 5,
    "api_version": "v5"
  },
  "action": {
    "id": 2116,
    "status": "in-progress",
    "user": "admin",
    "created_at": "2025-02-07T13:12:44.442950+00:00",
    "started_at": "2025-02-07T13:12:44.440924+00:00",
    "completed_at": null,
    "resource": "104",
    "resource_type": "SNP",
    "type": "delete_snapshot",
    "progress": 0
  }
}
```

Periodic Snapshots
Snapshots can also be scheduled to provide weekly protection for a server. To do this, you need to configure a cron on the target server.

Currently, only weekly schedules are supported with the following parameters:

Info
Maximum retention indicates the maximum number of **non-protected** snapshots allowed. If creating a new snapshot would exceed this number, the oldest non-protected snapshot is deleted first.

Configure a Periodic Snapshot
To set up a periodic snapshot, use the following request:

HTTP Request

```
POST /ecs/v2/servers/{servername}/crons
{
  "type": "snapshot_schedule",
  "config": {
    "typology": "weekly",
    "max_retention": 2,
    "day_of_week": 0
  }
}
```

HTTP Response

```json
{
  "status": "ok",
  "cron": {
    "name": "snapshot_schedule_ec200409",
    "schedule": "0 0 * * 0 (m/h/dM/MY/d) UTC",
    "enabled": true,
    "last_run_at": null,
    "infos": {
      "max_retention": 2,
      "day_of_week": 0
    },
    "last_run_status": null
  }
}
```

Note
The `schedule` attribute can be adjusted to run at any time between 00:00 and 06:00 UTC.

View the Active Configuration
To see the currently active configuration:

HTTP Request

```
```

```
GET /ecs/v2/servers/{servername}/crons/snapshot_schedule_{servername}
```

HTTP Response

```json
{
  "status": "ok",
  "cron": {
    "name": "snapshot_schedule_ec200409",
    "schedule": "0 0 * * 0 (m/h/dM/MY/d) UTC",
    "enabled": true,
    "last_run_at": null,
    "infos": {
      "max_retention": 2,
      "day_of_week": 0
    },
    "last_run_status": null
  }
}
```

Modify the Active Configuration
The active configuration can be changed without creating a new one:

HTTP Request

```
PATCH /ecs/v2/servers/{servername}/crons/snapshot_schedule_{servername}
{
  "enabled": true,
  "max_retention": 2,
  "day_of_week": 6
}
```

HTTP Response

```json
{
  "status": "ok",
  "cron": {
    "name": "snapshot_schedule_ec200409",
    "schedule": "0 0 * * 0 (m/h/dM/MY/d) UTC",
    "enabled": true,
    "last_run_at": null,
    "infos": {
      "max_retention": 2,
      "day_of_week": 0
    },
    "last_run_status": null
  }
}
```

Delete the Configuration
To remove the configuration and disable periodic snapshots, execute:

HTTP Request

```
DELETE /ecs/v2/servers/{servername}/crons/snapshot_schedule_{servername}
```

HTTP Response

```json
{
  "status": "ok",
  "cron": {
    "name": "snapshot_schedule_ec200409",
    "schedule": "0 0 * * 0 (m/h/dM/MY/d) UTC",
    "enabled": true,
    "last_run_at": null,
    "infos": {
      "max_retention": 2,
      "day_of_week": 0
    },
    "last_run_status": null
  }
}
```

==
========

DOC: hosting/cloudserver/rest-api/API-Endpoints/Templates/index.txt
----

Templates
Unlike Images, Templates are created by the user and are private.
They allow you to create images from servers already in use while
preserving:

- Users and their passwords
- Installed software
- Every file on the system

The following are **not** preserved:

- Network interfaces
- Their IPs and MAC addresses

Create a Template
To create a new template the following request is available:

HTTP Request
```http
POST /ecs/v2/templates
```

```
{
  "server": "ec000001",          // from which Server to create the
template
  "description": "my first template" // a short description for the
template
}
```

HTTP Response
```json
{
  "status": "ok",
  "action_id": 700,
  "template": {
    "id": 2,
    "name": "ei000001",
    "creation_date": "2019-04-18T12:34:08+00:00",
    "active_flag": false,
    "status": "CG",
    "uuid": "6708cdc4-0c8d-4ccf-83c7-ae39bccaeb42",
    "description": "my first template",
    "notes": "",
    "public": false,
    "cloud_image": false,
    "so_base": "debian",
    "required_disk": 20,
    "api_version": "v5",
    "api_version_value": 5,
    "version": ""
  }
}
```

You might receive the following errors:

Clone from a Snapshot
In addition to cloning from a currently active server, you can generate a Template directly from a Snapshot. To do so, just modify the previous call slightly:

HTTP Request
```http
POST /ecs/v2/templates
{
  "snapshot": 104,               // from which Snapshot to create the
template
  "description": "my first template" // a short description for the
template
}
```

HTTP Response
```json
{
```

```
  "status": "ok",
  "action_id": 700,
  "template": {
    "id": 2,
    "name": "ei000001",
    "creation_date": "2019-04-18T12:34:08+00:00",
    "active_flag": false,
    "status": "CG",
    "uuid": "6708cdc4-0c8d-4ccf-83c7-ae39bccaeb42",
    "description": "my first template",
    "notes": "",
    "public": false,
    "cloud_image": false,
    "so_base": "debian",
    "required_disk": 20,
    "api_version": "v5",
    "api_version_value": 5,
    "version": ""
  }
}
```

List All Templates
To list all templates the following request is available:

HTTP Request
```http
GET /ecs/v2/templates
```

HTTP Response
```json
{
  "status": "ok",
  "templates": [
    {
      "id": 2,
      "name": "ei000002",
      "creation_date": "2019-04-19T12:34:08+00:00",
      "active_flag": true,
      "status": "CD",
      "uuid": "6708cdc4-0c8d-4ccf-83c7-ae39bccaeb42",
      "description": "my second template",
      "notes": "",
      "public": false,
      "cloud_image": false,
      "so_base": "debian",
      "required_disk": 20,
      "api_version": "v5",
      "api_version_value": 5,
      "version": ""
    },
    {
      "id": 1,
```

```
      "name": "ei000001",
      "creation_date": "2019-04-18T12:34:08+00:00",
      "active_flag": true,
      "status": "CD",
      "uuid": "8520asd5-1k6l-6hjq-67g9-rf46rfvbgt97",
      "description": "my first template",
      "notes": "",
      "public": false,
      "cloud_image": false,
      "so_base": "centos",
      "required_disk": 20,
      "api_version": "v5",
      "api_version_value": 5,
      "version": ""
    },
    ...
  ]
}
```

Retrieve a Specific Template
To retrieve a specific template the following request is available:

HTTP Request
```http
GET /ecs/v2/templates/{templatename}
```

HTTP Response
```json
{
  "status": "ok",
  "template": {
    "id": 2,
    "name": "ei000002",
    "creation_date": "2019-04-19T12:34:08+00:00",
    "active_flag": true,
    "status": "CD",
    "uuid": "6708cdc4-0c8d-4ccf-83c7-ae39bccaeb42",
    "description": "my second template",
    "notes": "",
    "public": false,
    "cloud_image": false,
    "so_base": "debian",
    "required_disk": 20,
    "api_version": "v5",
    "api_version_value": 5,
    "version": ""
  }
}
```

Rename a Template
To rename a template you can execute the following request:

HTTP Request
```http
PUT /ecs/v2/templates/{templatename}
{
  "description": "my new name"
}
```

HTTP Response
```json
{
  "status": "ok",
  "template": {
    "id": 2,
    "name": "ei000002",
    "creation_date": "2019-04-19T12:34:08+00:00",
    "active_flag": true,
    "status": "CD",
    "uuid": "6708cdc4-0c8d-4ccf-83c7-ae39bccaeb42",
    "description": "my new name",
    "notes": "",
    "public": false,
    "cloud_image": false,
    "so_base": "debian",
    "required_disk": 20,
    "api_version": "v5",
    "api_version_value": 5,
    "version": ""
  }
}
```

Delete a Template
To delete a template you can use the following request:

HTTP Request
```http
DELETE /ecs/v2/templates/{templatename}
```

HTTP Response
```json
{
  "status": "ok",
  "action": {
    "id": 709,
    "status": "in-progress",
    "user": "foo",
    "created_at": "2024-10-19T15:08:55.887799+00:00",
    "started_at": "2024-10-19T15:08:55.887284+00:00",
    "completed_at": "2024-10-19T15:09:56.067592+00:00",
    "resource": "ei000001",
    "resource_type": "DI",
```

```
    "type": "delete_template",
    "progress": 10
  }
}
```

Warning
Once a template is removed, all data inside it will be permanently deleted.

==
========

DOC: hosting/cloudserver/rest-api/API-Endpoints/index.txt

Overview
Cloud Server Shared CPU is the on-demand cloud service ideal for testing, development, staging, and for anyone who needs to create and destroy cloud machines with consumption-based billing, quickly and at optimized costs.

In this section the Rest API functionalities of the following services are detailed:

- Cloud Server Shared CPU
- Cloud Server High Memory
- Cloud Server GPU

Create Your First Server via API
Creating a virtual server via API is very simple, and this page contains all the information needed to instantiate a virtual server quickly.

In this guide the command-line tool **curl** is used to perform the calls, but any HTTP client can be used instead without issues, such as **Postman** or **Bruno**.

Perform the login
... (the rest of the guide would continue)
First operation
You need to log in to our services and generate an access JWT token. An explanation of what a JWT token is is outside the scope of this guide and is not required for using our services. It can be thought of as a key that identifies who generated it and who uses it.

To generate it, execute the following request:

```bash
curl -X POST https://api.seeweb.it/ecs/v2/login \
-H 'Content-Type: application/json' \
-d '{
  "username": "*your username*",
  "password": "*your password*"
}'
```

```

**The response (if the credentials are correct) will be similar to the following:**

```json
{
  "token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJtc2ciOiJXcm9uZyBQbGFjZSBkdWRlPyJ
9.vBb9pIUYaxiGtO-7HY6hzBXH-q7L0GJS8stz2VYqb90",
  "expire": 1728560902,
  "status": "ok"
}
```

The token in the response body is the user's JWT token, and the **expire** field indicates its expiration time.

If you want to know when the token will no longer be valid, you can copy the **expire** value and paste it into a Unix Timestamp Converter website, or simply use the following calculator:

> **Info**
> For more information on user management, see **Account**.
> You can also use API-Tokens to authenticate to our services; see **API Token** for details.

Create a Server

Once you have the JWT token, you can proceed to create a server. For testing, we will create a small Ubuntu server in the Frosinone data center.

Get Available Plans

The first step is to list and identify the plan you are interested in. Do it with the following request:

```bash
curl -X GET https://api.seeweb.it/ecs/v2/plans \
-H 'Authorization: Bearer *your token*'
```

Sample response:

```json
{
  "status": "ok",
  "plans": [
    {
      "id": 1,
      "name": "eCS1",
```
```

```
      "cpu": "1",
      "ram": "1024",
      "disk": "20",
      "gpu": "0",
      "gpu_label": null,
      "hourly_price": 0.019,
      "montly_price": 14.0,
      "windows": false,
      "host_type": "ECS",
      "available": true,
      "available_regions": [
        { "id": 2, "location": "it-mi2", "description": "Milan" },
        { "id": 3, "location": "it-fr2", "description": "Frosinone" },
        { "id": 6, "location": "ch-lug1", "description": "Lugano" },
        { "id": 7, "location": "bg-sof1", "description": "Sofia" }
      ]
    },
    {
      "id": 7,
      "name": "eCS2",
      "cpu": "2",
      "ram": "2048",
      "disk": "40",
      "gpu": "0",
      "gpu_label": null,
      "hourly_price": 0.032,
      "montly_price": 23.0,
      "windows": false,
      "host_type": "ECS",
      "available": true,
      "available_regions": [
        { "id": 2, "location": "it-mi2", "description": "Milan" },
        { "id": 3, "location": "it-fr2", "description": "Frosinone" },
        { "id": 6, "location": "ch-lug1", "description": "Lugano" }
      ]
    }
  ]
}
```


Get Available Regions

Having identified the plan of interest (in our case **eCS1**), we can check its availability in the various data centers:

```bash
curl -X POST https://api.seeweb.it/ecs/v2/regions/availables \
-H 'Content-Type: application/json' \
-H 'Authorization: Bearer *your token*' \
-d '{
  "plan": "eCS1"
}'
```

Sample response:

```json
{
  "status": "ok",
  "regions": [
    [
      "ch-lug1",
      "it-fr2",
      "it-mi2",
      "bg-sof1"
    ]
  ]
}
```

Thus, the **eCS1** plan can be deployed in the data center identified as **it-fr2**, located in Italy (Frosinone).

Get Available Operating Systems

Next, identify an operating system you want to use:

```bash
curl -X POST https://api.seeweb.it/ecs/v2/images/basics \
-H 'Content-Type: application/json' \
-H 'Authorization: Bearer *your token*'
```

Sample response:

```json
{
  "status": "ok",
  "images": [
    {
      "id": 408,
      "name": "debian-9",
      "creation_date": "2019-10-02T16:48:18+00:00",
      "active_flag": true,
      "status": "CD",
      "uuid": "a1fa433a-ef1a-4b09-8571-514474847495",
      "description": "Debian Linux 9",
      "notes": "Debian Linux distribution",
      "public": true,
      "cloud_image": false,
      "so_base": "debian",
      "api_version": "v4",
      "version": "9"
    },
    {
      "id": 409,
      "name": "ubuntu-1604",
      "creation_date": "2019-10-02T16:48:18+00:00",
```

```
      "active_flag": true,
      "status": "CD",
      "uuid": "87098cfe-9e2c-4020-acd3-c91e46fe46a9",
      "description": "Ubuntu Linux 16.04 LTS",
      "notes": "Ubuntu Linux distribution",
      "public": true,
      "cloud_image": false,
      "so_base": "ubuntu",
      "api_version": "v4",
      "version": "1604"
    },
    {
      "id": 509,
      "name": "ubuntu-2204",
      "creation_date": "2022-09-02T12:38:46+00:00",
      "active_flag": true,
      "status": "CD",
      "uuid": "2d41e9c4-280d-423b-8110-9d14173f601c",
      "description": "Ubuntu Linux 22.04 LTS",
      "notes": "Ubuntu Linux 22.04 LTS",
      "public": true,
      "cloud_image": false,
      "so_base": "ubuntu",
      "api_version": "v4",
      "version": "2204"
    },
    {
      "id": 411,
      "name": "windows-2012r2",
      "creation_date": "2019-10-02T16:48:18+00:00",
      "active_flag": true,
      "status": "CD",
      "uuid": "618c4587-812e-4e9c-bb1c-c7976f671477",
      "description": "Windows Server 2012 R2",
      "notes": "Windows Server 2012 R2 Standard Edition",
      "public": true,
      "cloud_image": false,
      "so_base": "windows",
      "api_version": "v4",
      "version": "2012"
    }
  ]
}
```


If you want to use **Ubuntu Linux 22.04 LTS**, the code you need is **ubuntu-2204**.

Create a New Server

Finally, create the server:

```bash
curl -X POST https://api.seeweb.it/ecs/v2/servers \
```

```
-H 'Content-Type: application/json' \
-H 'Authorization: Bearer *your token*' \
-d '{
  "plan": "eCS1",
  "image": "ubuntu-2204",
  "location": "it-fr2",
  "notes": "my-first-server"
}'
```

Sample response:

```json
{
  "status": "ok",
  "action_id": 37430,
  "server": {
    "name": "ec205827",
    "ipv4": "",
    "ipv6": "",
    "group": null,
    "plan": "eCS1",
    "plan_size": {
      "core": "1",
      "ram": "1024",
      "disk": "20",
      "gpu": "0",
      "gpu_label": null,
      "host_type": "ECS"
    },
    "reserved_plans": [],
    "is_reserved": false,
    "reserved_until": "",
    "support": null,
    "location": "it-fr2",
    "location_label": "Frosinone",
    "notes": "my-first-server",
    "so": "ubuntu-2204",
    "so_label": "Ubuntu Linux 22.04 LTS",
    "creation_date": "2024-10-24T14:43:13.150586+00:00",
    "deletion_date": null,
    "active_flag": false,
    "status": "Booting",
    "progress": 0
  }
}
```
```json
"api_version": "v4",
"user": "eCS00099"
}
}
```

Monitoring Creation
To monitor the creation of the server you can use the `action_id` field received in the previous request (in the example: `37430`).
You can do it with the following request:

```bash
curl -X GET https://api.seeweb.it/ecs/v2/actions/37430 \
-H 'Authorization: Bearer *your token*'
```

The response will be similar to the following:

```json
{
  "status": "ok",
  "action": {
    "id": 37430,
    "status": "in-progress",
    "user": "eCS00099",
    "created_at": "2024-10-24T14:43:13.611722+00:00",
    "started_at": "2024-10-24T14:43:13.611449+00:00",
    "completed_at": null,
    "resource": "ec205827",
    "resource_type": "ECS",
    "type": "create_server",
    "progress": 25
  }
}
```

You can monitor the action until the status becomes **`completed`**:

```json
{
  "status": "ok",
  "action": {
    "id": 37430,
    "status": "completed",
    "user": "eCS00099",
    "created_at": "2024-10-24T14:43:13.611722+00:00",
    "started_at": "2024-10-24T14:43:13.611449+00:00",
    "completed_at": "2024-10-24T14:52:13.611449+00:00",
    "resource": "ec205827",
    "resource_type": "ECS",
    "type": "create_server",
    "progress": 100
  }
}
```

Viewing the Server
Now you can view the created server using the following request:

```bash
```

```
curl -X GET https://api.seeweb.it/ecs/v2/servers/ec205827 \
-H 'Authorization: Bearer *your token*'
```

And you will get:

```json
{
  "status": "ok",
  "server": {
    "name": "ec205827",
    "ipv4": "213.171.185.95",
    "ipv6": "2001:4b7e:0100:025d::1",
    "group": null,
    "plan": "eCS1",
    "plan_size": {
      "core": "1",
      "ram": "1024",
      "disk": "20",
      "gpu": "0",
      "gpu_label": null,
      "host_type": "ECS"
    },
    "reserved_plans": [],
    "is_reserved": false,
    "reserved_until": "",
    "support": null,
    "location": "it-fr2",
    "location_label": "Frosinone",
    "notes": "my-first-server",
    "so": "ubuntu-2204",
    "so_label": "Ubuntu Linux 22.04 LTS",
    "creation_date": "2024-10-24T14:51:03.360567+00:00",
    "deletion_date": null,
    "active_flag": true,
    "status": "Booted",
    "progress": 100,
    "api_version": "v4",
    "user": "seeweb_test",
    "virttype": "KVM"
  }
}
```

What to Do If the Server Status Is **`Customizing`**
If you check the server's `status` right after the action is marked as `completed`, the server might still be in the **`Customizing`** state. This state indicates that the server has been created but a customization process is still running on it, so some functions may not be available yet. Just wait a few minutes until the status changes to **`Booted`**.

DOC: hosting/serverless/billing/index.txt

Consumption
On the **Consumption** page you can see the monthly usage of our PODs. To do this, click **Consumption** in the left-hand menu of the panel or go directly to the Consumption page.
The following page will open:

Exporting Consumption
You can also export the consumption data shown in the panel as a CSV file. Simply select the month you are interested in and click **EXPORT CSV**; the file download will start automatically.

DOC: hosting/serverless/events/index.txt

Events
When POD usage is intensive, you may need to monitor their state changes over time. For this, the **Events** page is available.
To access it, click **Events** in the left-hand menu of the panel or go directly to Events.
A page similar to the one below will open:

On this page you can search events by POD, Namespace, or Cluster and scroll through the different events.

Tip
Event pagination works the same way as it does for our Cloud Server services. For more information, see **Cloud Server Actions**.

DOC: hosting/serverless/how-to-use/index.txt

How to Integrate Serverless Into a K8s Cluster
After logging into the panel and obtaining the kubeconfig for your tenant, you can integrate the GPU serverless services into your private cluster.

Install the Chart
To use our services you need to install our Helm chart on your cluster.
Run the following commands, replacing:

- `{tenant name}` with the desired tenant name
- `{path to your tenant kubeconfig}` with the absolute path to your kubeconfig

```bash
helm repo add clastix https://clastix.github.io/charts
helm repo update
helm upgrade --install k8sgpu clastix/k8sgpu \
  --namespace kube-system \
  --set "k8sgpuOptions.tenantName={tenant name}" \
  --set-file "kubeConfigSecret.content={path to your tenant kubeconfig}"
```

```

### Verify the GPU Node
After installing the chart, the `k8s.gpu` node should appear in your cluster:

```bash
kubectl get nodes
```

```
NAME        STATUS   ROLES   AGE   VERSION
k8s.gpu     Ready    agent   37m   v1.0.0
...
```

Check Available RuntimeClasses
Once the node is available you can list all the RuntimeClasses that can be used:

```bash
kubectl get runtimeclasses
```

```
NAME                      HANDLER   AGE
seeweb-nvidia-1xa100    nvidia    13h
seeweb-nvidia-1xa30     nvidia    13h
seeweb-nvidia-1xl4      nvidia    13h
```

Info
A **RuntimeClass** is simply a set of GPU resources that can be made available to our PODs.

Use a RuntimeClass
After selecting the RuntimeClass you want, create your PODs specifying which RuntimeClass to use:

```bash
cat << EOF | kubectl create -f -
apiVersion: v1
kind: Pod
metadata:
  name: nvidia-smi
spec:
  restartPolicy: OnFailure
  runtimeClassName: seeweb-nvidia-1xa30
  #runtimeClassName: seeweb-nvidia-1xl4
  #runtimeClassName: seeweb-nvidia-1xa100
  containers:
  - name: nvidia
    image: nvidia/cuda:11.0.3-base-ubuntu20.04
    command: ["/bin/bash", "-c", "--"]
```
```

```
    args: ["sleep 3600"]
    imagePullPolicy: Always
EOF
```
Following toleration is required on some distributions, e.g. AKS
as no CNI is assigned on virtual nodes
tolerations:
- key: "node.kubernetes.io/network-unavailable"
operator: "Exists"
effect: "NoSchedule"
EOF

The POD should now appear on the virtual GPU node with the ability to access the GPU specified by the RuntimeClass:
```bash
kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
nvidia-smi  1/1     Running   0          37m
```

We can verify GPU access with the following command:
```bash
kubectl exec nvidia-smi -- nvidia-smi
+-----------------------------------------------------------------------------
-----------------+
| NVIDIA-SMI 550.54.15   Driver Version: 550.54.15   CUDA Version: 12.4
|
|-----------------------------------------+------------------------+-----
-----------------+
| GPU  Name        Persistence-M | Bus-Id        Disp.A | Volatile
Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap | Memory-Usage | GPU-Util  Compute M. |
|                                |              | MIG M.               |
|=========================================+========================+=====
=================|
| 0   NVIDIA A30   On                | 00000000:06:00.0 Off  | 0
|
| N/A  32C  P0   27W / 165W          | 0MiB / 24576MiB       | 0%
Default         |
|                                |              | Disabled
|
+-----------------------------------------+------------------------+-----
-----------------+
+-----------------------------------------------------------------------------
-----------------+
| Processes:
|
| GPU   GI   CI   PID   Type   Process name   GPU Memory
|
| ID    ID   Usage
|
|=============================================================================
=================|
```

```
| No running processes found
|
+-----------------------------------------------------------------------
-----------------+
```

Uninstall the Chart
To uninstall the Chart from your cluster, simply run the following commands:
```bash
helm uninstall k8sgpu --namespace kube-system
```

Using Serverless GPU with K3S
Communicating with GPU pods when using a K3S cluster may cause anomalies because K3S nodes use websocket tunnels to interface with the control plane.
To resolve these anomalies, simply disable this feature; to do so, follow these steps.

Modify the systemd service
As a first step, you need to make changes to the k3s systemd service. Inside the folder `/etc/systemd/system/k3s.service.d` you need to create the file `override.conf` with the following content:
```ini
[Service]
ExecStart=
ExecStart=/usr/local/bin/k3s server --egress-selector-mode=disabled
```

Reload the Service
After that, you need to reload the service with the following command:
```bash
sudo systemctl daemon-reload
sudo systemctl restart k3s
```

Current Limitations
K8sGPU Agent is a work-in-progress solution. Like many new things, it can always be improved. Please be patient and provide honest feedback to help us improve it. Currently, the following limitations exist (some will be removed in upcoming releases):

- Pods cannot mount local storage or CSI PersistentVolume
- Pods cannot access other local Kubernetes services
- Pods can only access S3 storage
- Pods cannot be exposed on the local cluster
- Pods can only be exposed on the public Internet and are accessed via the HTTPS endpoint in the form `https://<your_tenant_id>.k8sgpu.net`
- Remember to provide authentication

DOC: hosting/serverless/introduction/index.txt

Overview
Serverless GPU is a Cloud-Native and Kubernetes-ready solution that gives you access to remote GPUs, scaling them without limits. It makes AI code execution more efficient thanks to:

- Fully dedicated GPUs available on demand
- Maximum compute power for your AI workloads
- Multi-GPU provisioning in seconds
- Kubernetes-ready cloud that can be integrated with any pre-configured environment

It makes it possible to integrate your local Kubernetes cluster with our dedicated GPU nodes.
Through the open-source Kubernetes container technology, which enables running application processes on diverse and multiple platforms, development teams extend the capabilities of their on-premise or cloud infrastructure dynamically and transparently, seeing Seeweb GPUs as if they were local to their server.
You can thus expand the capacity of any Kubernetes cluster, wherever it is in the world, via a secure link.

DOC: hosting/serverless/pod/index.txt
Pods
To view Pods, you can click on **Pod** in the left sidebar of the panel or go to Pods.
A page similar to the following will appear:

From here you can view all active Pods and Pods terminated in the past 15 days.

Delete a POD
From this page you can also force the deletion of a POD; to do so, click the **DESTROY** button on the row of the POD in question.
After that, a deletion request will be sent and once processed the pod will transition to the **terminated** state.
Obviously, you can also remove the POD directly from the local cluster that uses it.

DOC: hosting/serverless/rest-api/billing/index.txt
Usage
Of course, via API you can also view usage; it can be done with the following request:

HTTP Request
```
GET /k8sgpu/v1/billing/{year}/{month}
```

HTTP Response

```json
{
  "month": 11,
  "year": 2024,
  "pods": [
    {
      "pod": {
        "cluster": "skg00018",
        "online": false,
        "local_reference_name": "69d416b2-1c02-4910-9fcb-ae0d0da408f0",
        "local_reference_namespace": "skg00018-gpu",
        "remote_reference_name": "inference-service-5cfb7489f4-2d4xj"
      }
    }
  ]
}
```
```json
"remote_reference_namespace": "remote-gpu"
},
"cost": 281.88,
"online_hour": 348,
"runtime_class": "seeweb-nvidia-1xa6000",
"active": true
},
{
"pod": {
"cluster": "skg00018",
"online": false,
"local_reference_name": "d7b1c84f-9441-43bd-820c-7a06ed2db094",
"local_reference_namespace": "skg00018-gpu",
"remote_reference_name": "inference-service-5cfb7489f4-5xvs9",
"remote_reference_namespace": "remote-gpu"
},
"cost": 281.88,
"online_hour": 348,
"runtime_class": "seeweb-nvidia-1xa6000",
"active": true
},
{
"pod": {
"cluster": "skg00018",
"online": false,
"local_reference_name": "fc291986-c413-43fd-bec4-ff298b35ecfe",
"local_reference_namespace": "skg00018-gpu",
"remote_reference_name": "inference-service-5cfb7489f4-vmlwb",
"remote_reference_namespace": "remote-gpu"
},
"cost": 281.88,
"online_hour": 348,
"runtime_class": "seeweb-nvidia-1xa6000",
"active": true
},
]
```

```
}
```

Export consumption in CSV
It is also possible to export your consumption directly in CSV; to do this you need to set your `Accept` header to:

```
accept: text/csv
```

The API will automatically format the data in the correct format and return the response text.

DOC: hosting/serverless/rest-api/cluster/index.txt

Tenant
Managing your Tenants is also possible via API in a fully automatable way.

Create a Tenant
To create a new Tenant you must use the following request:

HTTP Request

```
POST /k8sgpu/v1/cluster/
```

HTTP Response

```json
{
  "message": "Cluster in creation",
  "cluster": "skg100004"
}
```

List all Tenants
To list all active Tenants simply use the following request:

HTTP Request

```
GET /k8sgpu/v1/cluster/
```

Response

```json
[
  {
```

```
    "name": "skg00018",
    "label": "tenant-lab",
    "customer": "skgu00004",
    "active": true,
    "created": true,
    "created_at": "2024-08-12T14:46:49.127000+00:00"
  },
  {
    "name": "skg100004",
    "label": "tenant-prod",
    "customer": "skgu00004",
    "active": true,
    "created": true,
    "created_at": "2024-08-13T12:50:17.683000+00:00"
  }
]
```

Retrieve a specific Tenant
You can also retrieve a specific Tenant:

HTTP Request

```
GET /k8sgpu/v1/cluster/{cluster_name}
```

Response

```json
{
  "name": "skg100004",
  "label": "tenant-prod",
  "customer": "skgu00004",
  "active": true,
  "created": true,
  "created_at": "2024-08-13T12:50:17.683000+00:00"
}
```

Rename a Tenant
Each tenant has a label to make identification easier for the user; you can modify this label via API with the following request:

HTTP Request

```
PUT/PATCH /k8sgpu/v1/cluster/{cluster_name}
{
  "label": "my-new-label"
}
```

Response

```json
{
  "name": "skg100004",
  "label": "my-new-label",
  "customer": "skgu00004",
  "active": true,
  "created": true,
  "created_at": "2024-08-13T12:50:17.683000+00:00"
}
```

Obtain a KubeConfig for a specific Tenant
After creating a Tenant to access ServerlessGPU services, you need to generate a KubeConfig and install it on your cluster.
You can generate the kubeconfig via API with the following request:

HTTP Request

```
GET /k8sgpu/v1/cluster/{cluster_name}/kubeconfig
```

Delete a Tenant
It is also possible to delete a Tenant; this operation will terminate all PODs running on it. To do so, use the following request:

HTTP Request

```
DELETE /k8sgpu/v1/cluster/{cluster_name}
```

Response

```json
{
  "message": "Cluster in deletion",
  "cluster": "skg100026"
}
```

DOC: hosting/serverless/rest-api/events/index.txt

Events
Just like in the web console, you can search and view events via API.

Retrieve all events
To get all events, use the following request:

HTTP Request

```
GET /k8sgpu/v1/events/
```

Response

```json
{
  "items": [ // events found
    {
      "pod": "d64cc49a-1123-40e9-b995-e6fcd5b73a73",
      "cluster": "skg00018",
      "id": 1487,
      "uuid": "d64cc49a-1123-40e9-b995-e6fcd5b73a73",
      "user": "skgu00004",
      "date": "2024-11-21T04:37:38.959489+00:00",
      "runtime_class": "seeweb-nvidia-2xa6000",
      "local_reference_namespace": "skg00018-gpu",
      "local_reference_name": "07bd7071-846a-4518-bb9a-b8a755941a7c",
      "remote_reference_namespace": "remote-gpu",
      "remote_reference_name": "ops-remote-gpu-8fc4db74b-s4kqg",
      "event_epoch": 1732163922,
      "event_type": "started"
    },
    {
      "pod": "fead8b43-fc69-4331-981f-e578a414306e",
      "cluster": "skg00018",
      "id": 1486,
      "uuid": "fead8b43-fc69-4331-981f-e578a414306e",
      "user": "skgu00004",
      "date": "2024-11-21T04:32:05.057457+00:00",
      "runtime_class": "seeweb-nvidia-2xa6000",
      "local_reference_namespace": "skg00018-gpu",
      "local_reference_name": "259003c4-3d36-4411-ab9d-2be5a82ca087",
      "remote_reference_namespace": "remote-gpu",
      "remote_reference_name": "ops-remote-gpu-8fc4db74b-s4qng",
      "event_epoch": 1732163596,
      "event_type": "terminated"
    }
  ],
  "length": 2, // currently returned events
  "total": 95 // total existing events
}
```

Search events

This endpoint accepts a query parameter that allows you to search the events.

Add the `search` field to your request:

```
GET /k8sgpu/v1/events/?search=07bd7071-846a-4518-bb9a-b8a755941a7c
```

Paginate events
You can also paginate the results using the `offset` and `length` parameters:

```
GET /k8sgpu/v1/events/?offset=50&length=10
```

- **offset** - the index of the first event to return (skip the preceding events)
- **length** - the number of events to return after the offset

---

DOC: hosting/serverless/rest-api/intro/index.txt

Overview
(Translation of the introductory content would continue here…)
Obviously our Serverless GPU service exposes a public API to simplify access to information available from the panel to users' automated services.
The possible operations range from Account Management, to Tenant Management, up to Consumption export.

==

DOC: hosting/serverless/rest-api/pod/index.txt

Pod
Pod management is also available via API; however, creating a POD is not available either via API or via the panel, as the ServerlessGPU service does not aim to be a Kubernetes cluster management panel.
If you are looking for a management panel for your Kubernetes cluster, the Seeweb Team strongly recommends solutions such as Rancher.

View Your PODs
You can view all your PODs; to do this the following request is available:

HTTP Request
```
GET /k8sgpu/v1/pod/
```

HTTP Response
```json
[
  {
    "uuid": "ad5eabd2-396d-44c2-869e-c82a4df87f18",
    "remote_reference_name": "inference-service-5cfb7489f4-2d4xj",
    "remote_reference_namespace": "remote-gpu",
    "local_reference_name": "69d416b2-1c02-4910-9fcb-ae0d0da408f0",
```

```
    "local_reference_namespace": "skg00018-gpu",
    "status": "terminated",
    "online": false,
    "active": true,
    "succeeded": false,
    "failed": false,
    "creation_date": "2024-08-12T14:48:00+00:00",
    "deactivated_from": null,
    "runtime_class": "seeweb-nvidia-1xa6000",
    "user": "skgu00004",
    "cluster": "skg00018",
    "in_deletion": false
  },
  {
    "uuid": "18009a59-b1c8-4ab9-8ad5-a0c7c2efe6eb",
    "remote_reference_name": "inference-service-5cfb7489f4-5xvs9",
    "remote_reference_namespace": "remote-gpu",
    "local_reference_name": "d7b1c84f-9441-43bd-820c-7a06ed2db094",
    "local_reference_namespace": "skg00018-gpu",
    "status": "terminated",
    "online": false,
    "active": true,
    "succeeded": false,
    "failed": false,
    "creation_date": "2024-08-22T12:37:44+00:00",
    "deactivated_from": null,
    "runtime_class": "seeweb-nvidia-1xa6000",
    "user": "skgu00004",
    "cluster": "skg00018",
    "in_deletion": false
  }
]
```

Filter Your PODs by Tenant
You can filter your PODs by Tenant using the following query parameter:

```
GET /k8sgpu/v1/pod/?cluster=skg00018
```

View a Specific POD
Of course you can also view a specific POD:

HTTP Request
```
GET /k8sgpu/v1/pod/{pod uuid}
```

HTTP Response
```json
{
  "uuid": "ad5eabd2-396d-44c2-869e-c82a4df87f18",
  "remote_reference_name": "inference-service-5cfb7489f4-2d4xj",
```

```
  "remote_reference_namespace": "remote-gpu",
  "local_reference_name": "69d416b2-1c02-4910-9fcb-ae0d0da408f0",
  "local_reference_namespace": "skg00018-gpu",
  "status": "terminated",
  "online": false,
  "active": true,
  "succeeded": false,
  "failed": false,
  "creation_date": "2024-08-12T14:48:00+00:00",
  "deactivated_from": null,
  "runtime_class": "seeweb-nvidia-1xa6000",
  "user": "skgu00004",
  "cluster": "skg00018",
  "in_deletion": false
}
```

Delete a POD
As in the panel, you can also request POD deletion via API; to do this you must send the following request:

HTTP Request
```
DELETE /k8sgpu/v1/pod/{pod uuid}
```

==
========

DOC: hosting/serverless/rest-api/user-management/index.txt

User Management

Account
Various operations can be performed via API for managing user accounts, including login and retrieving account information.

Login
User operations are processed through our API, so you need to generate a JWT token to allow the user to identify themselves to our services. The following endpoint generates an access token from a username and password.

HTTP Request
```
POST /k8sgpu/v1/user/login/
{
  "username": "foo",
  "password": "foo-s3cr3t-passw0rd"
}
```

Response

```json
{
  "token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE3MzIxOTQyMzQsIm5iZiI6MTc
zMjE3OTgzNCwidXNlcm5hbWUiOiJhZG1pbiJ9.X-
fKDhNqkG3NNy0iTVHS1ZbPZ4odeGdeuRNJsnRKel4",
  "expire": "2024-11-21 13:03:54.736403",
  "user": "foo",
  "max_age": 14400
}
```

Info
The generated token can be used as a simple Bearer token until it
expires.

```bash
curl --request GET 'https://api.seeweb.it/k8sgpu/v1/' \
  --header 'Authorization: Your-JWT-Token'
```

Account Information
User information can be retrieved via the following endpoint:

HTTP Request
```
GET /k8sgpu/v1/user/
{
  "username": "foo"
}
```

Change Password
You can change a user's password directly through the API at this
endpoint:

HTTP Request
```
PUT/PATCH /k8sgpu/v1/user
{
  "password": "my-new-supers3cr3t-passw0rd"
}
{
  "username": "string"
}
```

==

DOC: hosting/serverless/tenant/index.txt

Tenant

To use the GPU nodes provided by the Serverless GPU service you must define one or more tenants; tenants represent a set of resources that can be accessed through the GPU nodes.

To view your tenants, click **“Tenant”** in the left-hand navigation bar of the panel.

Create a New Tenant
To create a new Tenant, simply click **“CREATE TENANT”** (1); a new Tenant in creation will be added to the tenant list.
The status will automatically become **ready** once the tenant is created on our remote service.

Generate a kubeconfig for a Specific Tenant
To create new resources in a given tenant using the GPU nodes supplied by Serverless GPU, you need to generate an access kubeconfig. Click **“KUBECONFIG”** (2).
The kubeconfig download will start automatically.

Rename a Tenant
To rename a Tenant, click the pencil icon next to the current Tenant name (4) and enter the desired name.

Destroy a Tenant
... (the original text ends here)
The destruction of a Tenant and all resources derived from it can be performed by clicking **“DESTROY”** (no. 3) and confirming the action.
Attention
Attention: The destruction of a Serverless GPU Tenant is irreversible.

==

DOC: hosting/serverless/user-management/index.txt

User Management
User management is one of the simplest operations from any part of the web panel via the top bar:
Just click on your username to open the dropdown menu for user management;
From there you can:

- Change your credentials
  To change your credentials, click **“Change Password”** within the account management. This will display the following form:
  In the form at the center of the page (no. 1) you can enter the new access credentials; the option to save the credentials will be enabled only if the requirements shown in the lower-right corner (no. 2) are fully met.

- Why should I log out?

Although our services use temporary JWT tokens, logging out is extremely important to secure our systems on your local application. Doing so makes our systems “forget” who your user is and forces any malicious person who may have taken over your terminal to log in again with credentials.

==

DOC: index.txt

Overview
Seeweb is the Italian Cloud Computing Provider that is part of the Dominion Hosting Holding group.
The first company to introduce cloud hosting services in Italy in 2009, opening the cloud era, Seeweb has also been a pioneer in the field of Artificial Intelligence and Machine Learning, with its GPU and NPU computing infrastructures in the cloud.
With its multiple owned data centers and the support of DHH points of presence, Seeweb facilitates the design and implementation of scalable, flexible, redundant IT architectures, with a minimum uptime of 99.9% backed by SLA-guaranteed penalties.

Its Housing and Colocation services allow businesses to place their servers and equipment on efficient infrastructures that balance performance and sustainability, thanks to a specific optimization of energy consumption recognized by ISO 14001 certification and acknowledged as a green provider by The Green Web Foundation.

A wide range of cloud services ensures service continuity and speed for management tools, databases, applications, and software of any type: our cloud servers provide horizontal and vertical scalability and a modularity that lets you flexibly follow the evolution of your projects. This includes AI and ML projects: in our GPU Cloud and NPU Server services you will find technology that combines top performance with maximum flexibility, with rapid, on-demand provisioning of compute resources.

Together with technology built from carefully selected top-tier vendors and the highest quality and sustainability of hardware components, Seeweb services are designed to serve complex, multicloud projects, supported by the Presales Engineering team and a Service Desk focused on delivering proactive and effective technical assistance.

Our IaaS and PaaS solutions are crafted to offer a simple user experience, free from any form of technological lock-in and integrable with any infrastructure thanks to technologies such as Kubernetes.

In the following guides, you will learn how to use our Cloud products and services to facilitate their daily use.