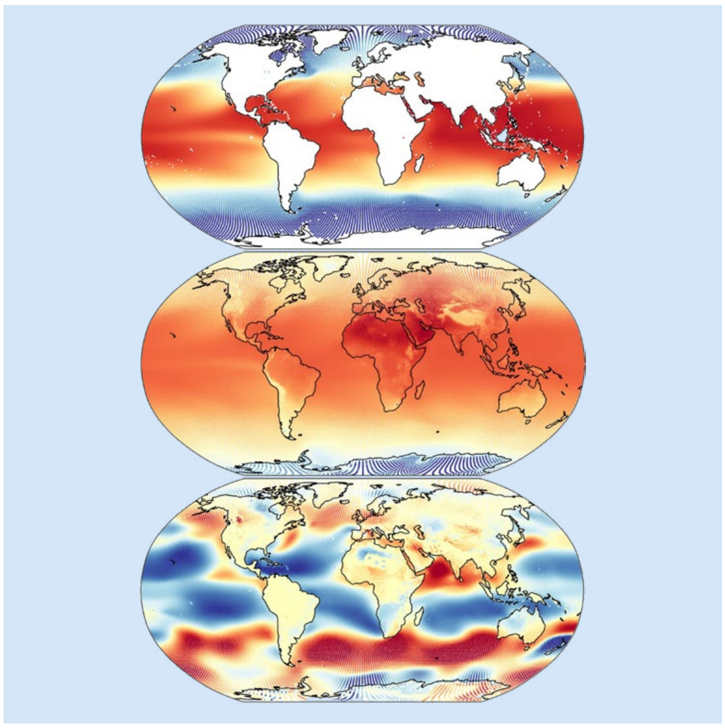
ECMWF Feature article

from Newsletter Number 182 – Winter 2024/25

COMPUTING

ECMWF Sites: websites as a service



www.ecmwf.int/en/about/media-centre/media-resources

doi: 10.21957/m5gl27f96m

This article appeared in the Computing section of ECMWF Newsletter No. 182 – Winter 2024/25, pp. 33–41

ECMWF Sites: websites as a service

Manuel Martins

ECMWF's web presence is pivotal not only in disseminating its forecast products, but in collaborating and communicating research findings to a diverse audience, including scientists, policymakers, and the general public.

ECMWF Sites is a service that enables ECMWF staff and visiting scientists to effortlessly create and publish websites, providing a basis for collaboration, communication and fast prototyping. It ensures appropriate control and communication about the quality of ECMWF websites or publications outside our main website, *www.ecmwf.int*, meeting a minimum quality bar for our web presence. *ECMWF Sites* provides mainly private websites for use by our scientists, but websites can also be made publicly available, subject to an approval process.

The platform is built using Kubernetes, an open-source system for automating the deployment, scaling, and management of containerised applications (*https://kubernetes.io*). It leverages the use of the Kubernetes operator pattern to manage the lifecycle of a website. The platform and all websites are hosted in the ECMWF data centre, running in Kubernetes on a dedicated virtual infrastructure.

This article presents the service and provides an overview of the platform's architecture along with insights on the past, present and future of the platform.

Background and inception

At ECMWF, collaboration is ingrained in our operational framework: scientists, researchers, and engineers work together, leveraging their diverse backgrounds to improve weather forecasting models and climate prediction tools. ECMWF actively collaborates with external entities, including national meteorological services, academic institutions, and other international organisations. Web-based collaboration tools play a crucial role in facilitating communication and cooperation. These platforms enable real-time data sharing, collaborative editing of documents, and the ability to provide feedback on ongoing projects, regardless of geographical location.

ECMWF Sites is a web platform that was planned with this in mind. It provides a simple way for users to create a space, manage content and share it with others, internally or externally. It was initially designed as an HTTP hosting service for static content, with simple and straightforward requirements:

- enough disk space
- easy content management
- accessible with or without authentication.

These requirements were gathered throughout the years from real use cases. Some simple solutions were previously available, but they were complex to use, creating confusion and frustration for users. These solutions also did not reflect a modern service-oriented way of working for users.

Designed and released to pre-production by the end of 2019, as a proof of concept on Kubernetes native applications, *ECMWF Sites* served as the experimental model for a controlled service fully integrated with Kubernetes. This offered a solid understanding of the platform and provided insights into how the team could leverage the Kubernetes platform at ECMWF, running production containers at scale.

The use of containers for production services was first implemented at ECMWF in mid-2018, with the re-architecture of the Atlassian suite of products. These products were deployed in Docker containers, on dedicated virtual machine infrastructure, managed by systemd and provisioned with Puppet.
We soon realised the need for a container orchestration platform. By the beginning of 2019, ECMWF was running the first production Kubernetes cluster and started to migrate small applications. A huge shift in technologies happened during this period, which meant some effort to ensure a smooth transition for all teams using the infrastructure.

ECMWF Sites was then released to production in April 2020, with less than 1,000 lines of Python code and around 25 websites published.

From the outset, the potential of the service was evident. Soon after, user feedback brought many additional requirements to light. The service became more customisable to the point that users can now build and run their own containerised applications as part of a website. Some of the requirements introduced later are:

- content retention control
- web analytics and reporting
- multi-factor authentication
- support for caching, web robots and cross-origin resource sharing (CORS)
- running custom applications (containers).

Almost five years later, *ECMWF Sites* is about 10,000 lines of code, and it is written in Python, Golang and Lua. At the time of writing, it hosts around 300 websites (see Figure 1).

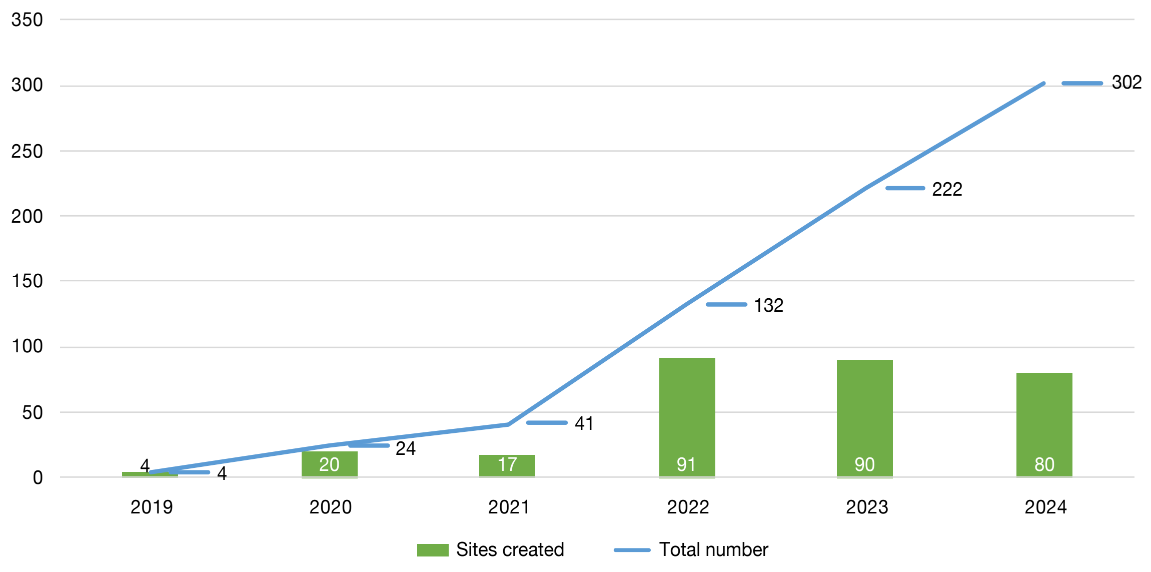


Figure 1 This chart shows the evolution of websites created and their overall number by the end of 2024.

Overview of the service

ECMWF Sites is a web platform accessible through *https://sites.ecmwf.int*. Accessing this sends the user to the *Sites Hub*, which serves as the central entry point, providing a list of all available websites, along with their details and URLs. Each website is part of a 'space', typically the user's username, and has a unique name within that space. Users can view and configure their websites through the Hub's site view page (accessible through the first column on *https://sites.ecmwf.int/hub/list/all/*). Websites are accessible via specific URLs, and administrators can manage all content through a web-based file browser or programmatically via a Representational State Transfer (REST) application programming interface (API).

Websites can be either private or public. Owners of private websites can share access with specific users or groups, who must authenticate to view the content. In addition to the web interface, users can interact with the platform using a Python Software Development Kit (SDK) or a Command Line Interface (CLI). A unique authentication token is generated for each website, enabling secure API interactions for automated tasks, without linking to specific user credentials.

Use cases

The service is being used for many different purposes by users with different backgrounds. Figures 2 and 3 show the distribution of websites per type and use case.

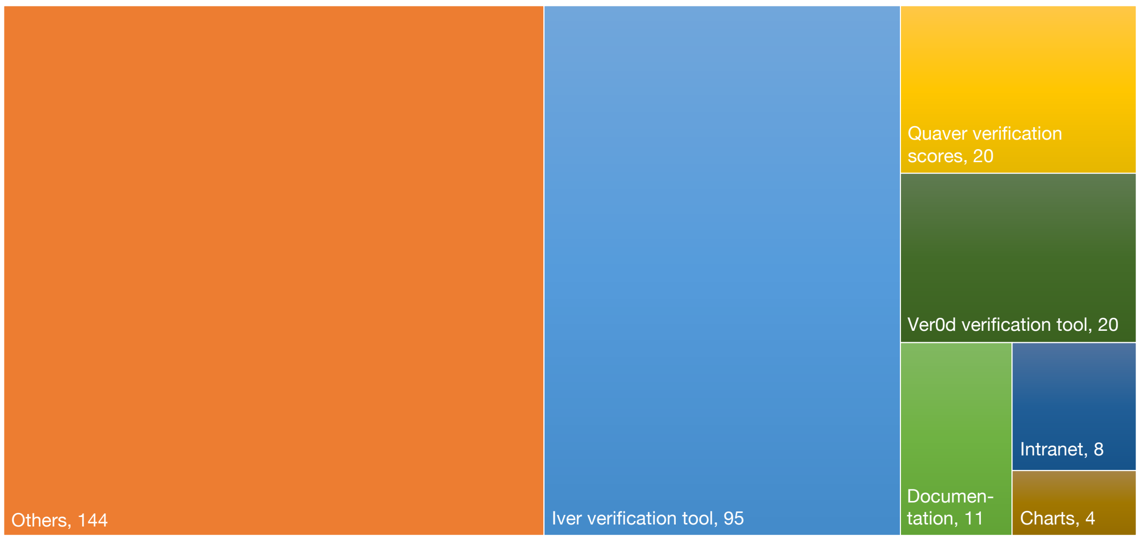


Figure 2 This chart shows the distribution of websites per use case at the end of 2024.

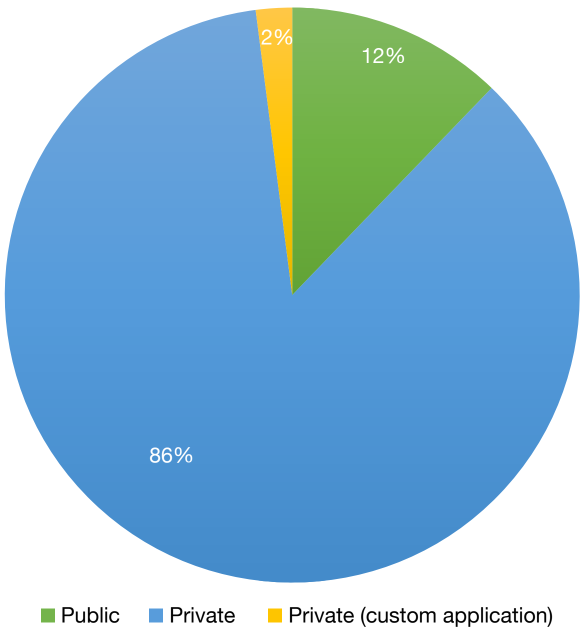


Figure 3 This chart shows the distribution of websites per type (private or public) at the end of 2024.

Verification and validation assessment plots are a very common use case of websites. The Iver private websites are used by scientists and researchers to access Iver verification tool results. This is done to assess the improvement their research provides over the current version of ECMWF's Integrated Forecasting System (IFS). During new IFS Cycle implementations, these websites are responsible for more than 1 TB of ingress traffic and 0.5 TB of egress traffic, per week. As ECMWF scientist Alan Geer describes:

> *"ECMWF Sites has revolutionised the way we share diagnostic plots with ECMWF scientists both within and outside ECMWF premises. My example is the Iver forecast verification tool, which has around 100 users across ECMWF and is one of the main ways in which the research department assesses the impact of upgrades to the IFS. When comparing the forecast and analysis quality of two experiments, Iver generates at least 14,000 plot panels, and these are summarised in a web page. Scientists run lots of experiments, meaning possibly hundreds of separate web pages. In the past,*

these web pages were only accessible within ECMWF, but now they are also accessible externally. To achieve this, each Iver user's workstation had to be turned into a mini web server to share the plots with others. Sites now deals with all of this and allows us to share these plots easily both inside and outside the organisation. It is comfortably hosting as many as 100 million plots from Iver users alone."

ECMWF Sites has been used as a fast-prototyping platform, allowing seamless integration with other web services. With its API-driven model, it allows easy uploads and it is often used as a repository for data, images, JavaScript and other static content, allowing other services to make these accessible, mainly to internal staff. As Helen Setchell, senior content architect and user experience coordinator at ECMWF, describes:

"As ECMWF's senior content architect and UX coordinator, I work with a variety of platforms to support our online presence, and while they each serve their purpose, they come with certain limitations – whether it's in development flexibility or optimising user experience. That's where Sites has made a real difference for ECMWF. We now have the freedom to create custom websites and experimental pages that break free from the constraints of our traditional platforms. It's given us a way to innovate without worrying about disrupting our core systems or forcing those platforms to do things they weren't designed for. One of the standout benefits is the ability to create solutions to share content seamlessly between platforms. This has allowed us to design user interfaces that truly cater to our audience's needs, instead of bending to the limitations of the platforms we use. Overall, ECMWF Sites has been a game-changer, offering the flexibility and control we've been looking for."

The *ECMWF Sites* platform supports many other public-facing web portals, such as https://pulse.climate.copernicus.eu, which uses data and charts constantly updated through the API. Furthermore, it is used to distribute monthly communication bulletins through media channels. As Julien Nicolas, a climate scientist at ECMWF, describes:

"Since 2023, ECMWF Sites has emerged as an indispensable tool for the C3S Climate Intelligence (CI) team, serving two key functions: file sharing with users external to ECMWF and back-end storage for web applications. For its monthly Climate Bulletins, the CI team relies on Sites to share a variety of graphics and data files. These are initially shared with web content collaborators (pre-publication) and later with journalists (post-publication). Additionally, each report includes embargoed content, for which access is securely restricted via password protection. ECMWF Sites has significantly streamlined this entire process, effectively replacing Dropbox, which was previously in use.
The second major application of sites within the CI team is file storage for web applications. A notable example is Climate Pulse, where all maps, CSV, and JSON files are stored and updated daily using the sites API, driven by an ecFlow suite. This ensures that data is kept up to date seamlessly. The CI team's overall experience with ECMWF Sites has been excellent. Its combination of a file browser and API offers great flexibility, supporting both manual and automated workflows. Based on our feedback, several enhancements were introduced, making the platform even more user friendly."

Custom Applications is a feature introduced in early 2022 and allows users to run a Docker container as part of their website. This is useful when users want to run some processing as part of their website, or access internal data from other systems. As Martin Janousek, an analyst at ECMWF, describes:

"I very much welcomed the implementation of ECMWF Sites as I had been calling for some sort of internal HTTP server to present various raw HTML documents since the early 2010s. Before ECMWF Sites, sharing of HTML documents was rather cumbersome. For example, one important product of every new model cycle evaluation, the scorecard, made as an HTML file with included JavaScript objects, was distributed either as an email attachment or put to a shared directory. That was rather impractical as staff started to work increasingly remotely. With ECMWF Sites a scorecard is shared as just a URL, conveniently accessible in users' local browsers. ECMWF Sites became even more useful when it offered an option to create dynamic applications. Step-by-step instructions and support from the team on how to build, test and deploy a docker container to ECMWF Sites were very helpful. It is also very important that, although website content can be conveniently accessed from anywhere, editing is managed by ECMWF Sites, limiting it to internal staff. I consider ECMWF Sites to be a key asset of ECMWF's IT systems, and I use it as a go-to solution for the implementation of future applications, in particular for data visualisation and access."

A wide range of websites is available and can be accessed at https://sites.ecmwf.int/hub/list/all/.

Architecture

This section details the architecture of the main components of the system. With reference to the schematics shown in Figure 4, the system is divided into four main components: *Hub*, *Operator*, *Site*, and *Jobs*.

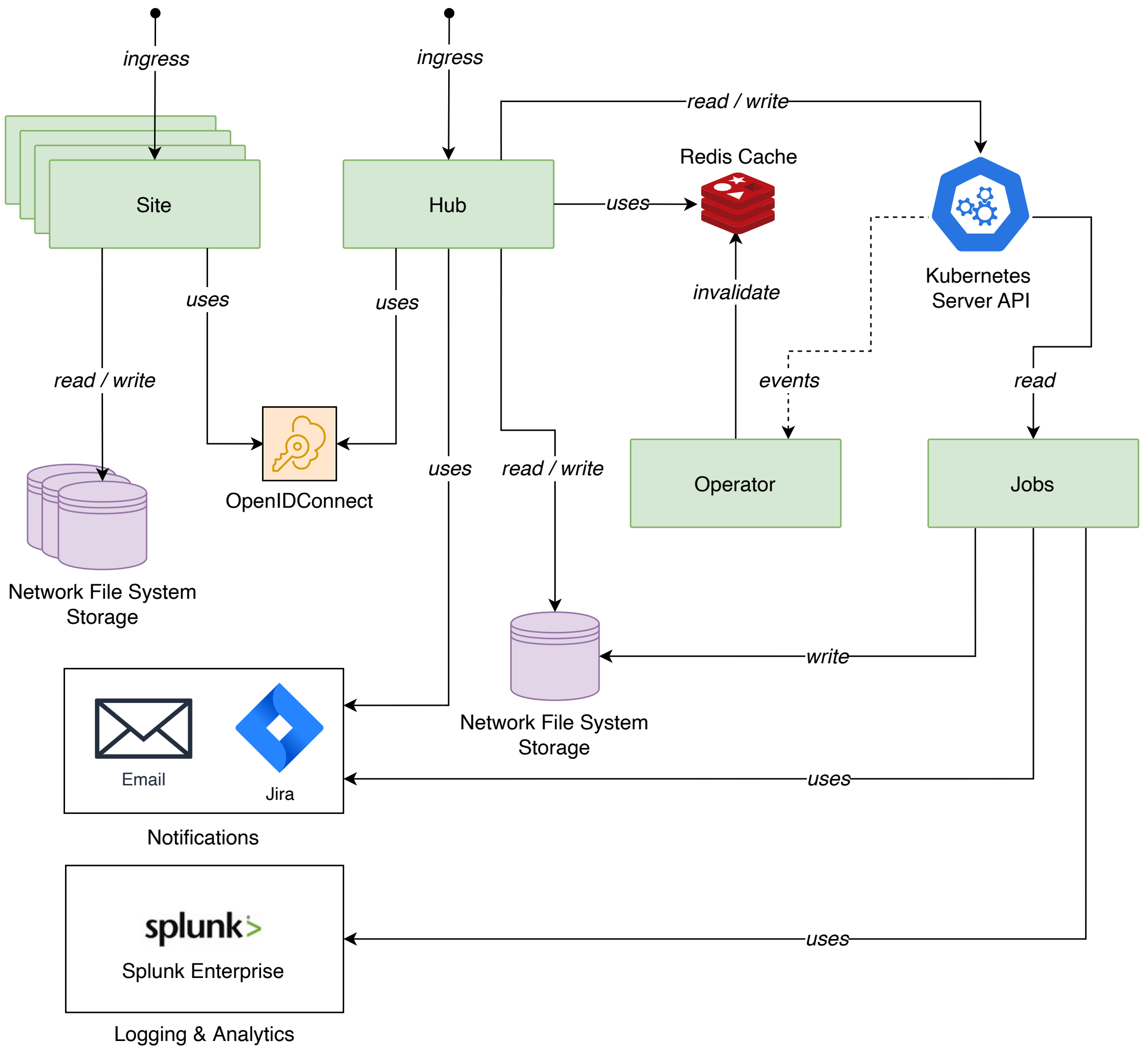
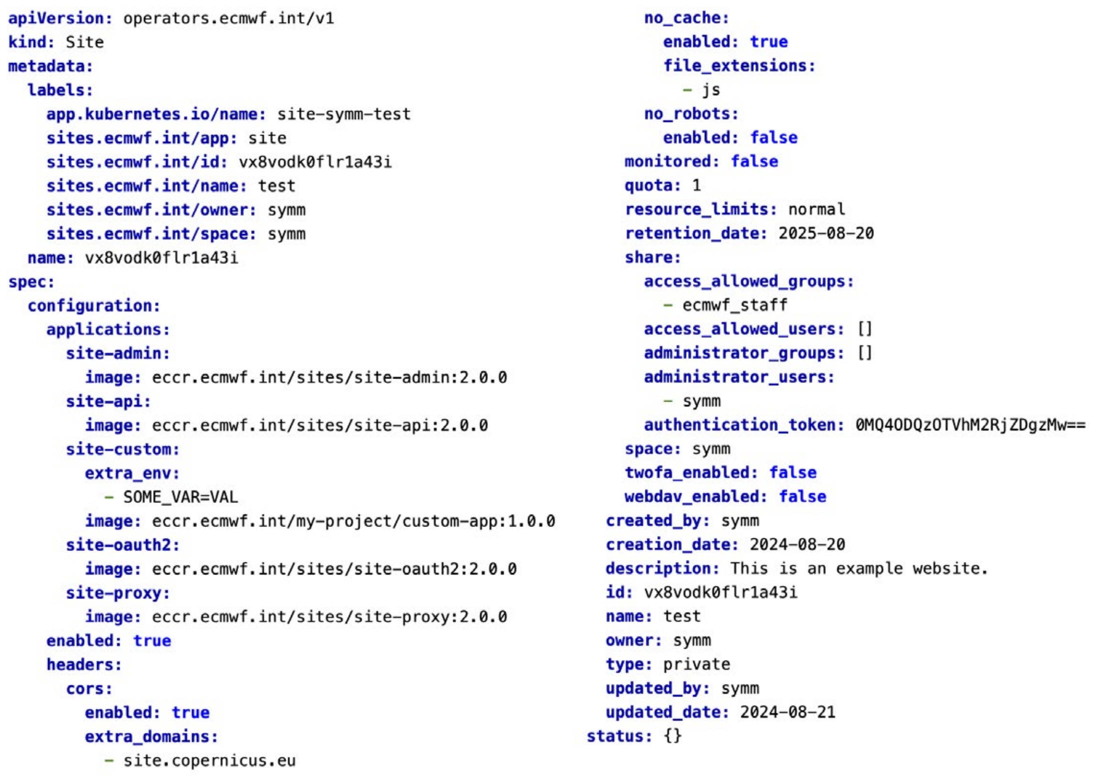


Figure 4 This diagram shows the high-level architecture of the platform with all the interactions between internal and external components.

The *Hub* component allows users to interact with the system through a web interface or a REST API. It is essentially an enhanced proxy to the Kubernetes API server. It aggregates all the websites from all the users and allows users not only to explore the available websites, but to manage their own websites. Users can also access all sorts of information, such as configurations, server logs and web analytics. The *Hub* is open to all users with an ECMWF account, but only ECMWF staff and visiting scientists are configured to create websites. The *Sites Hub* is accessible through:

- Web UI (*https://sites.ecmwf.int/hub/*)
- REST API (*https://sites.ecmwf.int/hub/api/v1/spec/*).

The *Operator* component is the most important of the four. This component follows the operator pattern, which is a design approach that extends the functionality of Kubernetes. It does so by automating the management of complex applications, acting as a custom controller encapsulating the operational knowledge needed to deploy, manage, and scale the applications. Specifically, this component is responsible for ensuring the desired state of each individual website. A CRD (Custom Resource Definition) in Kubernetes is a way to extend the Kubernetes API by defining custom resources. The *Sites Operator* takes these Site CRD events and makes sure the necessary components composing a website are aligned with the CRD configuration. These resources allow the management of new types of objects, specific to an application or infrastructure needs, just like the built-in Kubernetes objects (like Pods, Services, etc.). Figure 5 shows an example of a Site CRD object.



```yaml
apiVersion: operators.ecmwf.int/v1
kind: Site
metadata:
  labels:
    app.kubernetes.io/name: site-symm-test
    sites.ecmwf.int/app: site
    sites.ecmwf.int/id: vx8vodk0flr1a43i
    sites.ecmwf.int/name: test
    sites.ecmwf.int/owner: symm
    sites.ecmwf.int/space: symm
  name: vx8vodk0flr1a43i
spec:
  configuration:
    applications:
      site-admin:
        image: eccr.ecmwf.int/sites/site-admin:2.0.0
      site-api:
        image: eccr.ecmwf.int/sites/site-api:2.0.0
      site-custom:
        extra_env:
          - SOME_VAR=VAL
        image: eccr.ecmwf.int/my-project/custom-app:1.0.0
      site-oauth2:
        image: eccr.ecmwf.int/sites/site-oauth2:2.0.0
      site-proxy:
        image: eccr.ecmwf.int/sites/site-proxy:2.0.0
    enabled: true
    headers:
      cors:
        enabled: true
        extra_domains:
          - site.copernicus.eu
      no_cache:
        enabled: true
        file_extensions:
          - js
      no_robots:
        enabled: false
    monitored: false
    quota: 1
    resource_limits: normal
    retention_date: 2025-08-20
    share:
      access_allowed_groups:
        - ecmwf_staff
      access_allowed_users: []
      administrator_groups: []
      administrator_users:
        - symm
      authentication_token: 0MQ4ODQzOTVhM2RjZDgzMw==
    space: symm
    twofa_enabled: false
    webdav_enabled: false
  created_by: symm
  creation_date: 2024-08-20
  description: This is an example website.
  id: vx8vodk0flr1a43i
  name: test
  owner: symm
  type: private
  updated_by: symm
  updated_date: 2024-08-21
status: {}
```

Figure 5 A sample Site CRD object definition.

The *Site* component is essentially a set of built-in Kubernetes objects that, when composed, will host and expose an individual website. These objects are specified by an individual Site CRD, and its lifecycle is managed by the *Sites Operator*. The Kubernetes objects that compose a website are:

1. One *StatefulSet* running four container applications, or five when users specify a custom application, where each of these applications is responsible for a specific task:
   a. Oauth2 Proxy – responsible for the authentication part and integrated with ECMWF's authentication system. Will ensure users are logged in on private websites.
   b. Nginx Proxy (OpenResty) – responsible for proxying requests accordingly and serving files as part of the web server. It is responsible for the authorisation as well, ensuring only configured users can access private websites.
   c. Admin File Browser – this is the administration File Browser, which allows administrators to manage website contents.
   d. Admin REST API – this is the administration REST API, which also allows administrators to manage website contents.
   e. Custom Application – if users specified their own application to run as part of the website, then this will be served as the main application. It will not replace the Nginx Proxy. Instead, it will be proxied by the Nginx Proxy.
2. Two *ConfigMaps*, one with the global website configurations as environment variables and another with configuration files.
3. One Secret containing sensitive information to be used for authentication and authorisation purposes.

4. One *PersistentVolumeClaim* with the storage amount requested.
5. One Ingress to access the website through the main domain on a specific path: https://sites.ecmwf.int/*space/name/*

The *Jobs* component is composed by a set of packages to execute global administrative tasks such as: configurations backup, web analytics pre-loading, website storage quota, and retention date expiration notifications. These are essentially *CronJobs* running on intervals on the system.

ECMWF Sites is integrated with a few other systems to provide some of its functionality.

- All the components that are accessible to users, *Site* and *Hub*, use OpenIDConnect for authentication and authorisation.
- Network File System (NFS) volumes from TrueNAS are used as storage, and each individual website has its own volume.
- Redis is used for caching the *Hub* data, allowing users to interact with the system without overloading the Kubernetes Server API.
- Splunk is used for ingestion and extraction of log information, which is then used to produce web analytics.
- Jira and Email are used for notifications, such as when a website retention date expires, or a website storage is almost full, or to request access to a website, etc.

Security considerations

In order to ensure a high degree of compliance with security best practices, *ECMWF Sites* implements various strategies to minimise the attack surface of the platform and of other internal systems, and to prevent leakage of information.

In terms of infrastructure, the platform runs on a Kubernetes cluster which is deployed on Virtual Infrastructure within a specific network security zone. From within this security zone, it is only possible to access a set of standard ports running on other specific security zones, where other forecasting services run. Even though our internal Network Architecture has a high degree of segmentation and segregation, a set of *NetworkPolicies* further enforces internal traffic rules, blocking all traffic connections from and into each website, other namespaces and components within the Kubernetes cluster. This provides internal isolation between websites, and between other services running within the same Kubernetes cluster.

The Kubernetes API server is only accessible internally, from within a set of VLANs, within the ECMWF network. A *ServiceAccount* with a token is created for the internal communication between the *Hub* and *Operator* with the Kubernetes cluster server API. Role Based Access Control (RBAC) is then configured for this *ServiceAccount*, allowing access to a few cluster-wide resources for the management of the system. The *ServiceAccount* token is not mounted on individual websites, meaning custom applications will not be able to use it to exploit the system. It is worth noting that the aforementioned *NetworkPolicies* would block this traffic anyway.

In terms of authentication and authorisation, all endpoints use OpenIDConnect for authentication, relying on the JWT *access_token*, using the *preferred_username* and *entitlements* for authorisation.
A 64 characters Hex token is randomly generated for each website. This token is used as an administration token and allows anyone holding it to manage all the contents of the website. It is useful for automated pipelines, and in case tokens are compromised they can be revoked by generating new ones. Only one token is usable at any time for a specific website, until it is revoked.

CPU and Memory are limited by a *ResourceQuota* object. This ensures a global limit for resources that the platform can take from the Kubernetes cluster. Every website is configured with a sensible number of resources that can be used. These are configurable per website within three resource levels, defined as Normal, Medium, and High. This is very relevant for websites running custom applications, since these run code out of our control; we can limit and protect the system from misbehaving containers:

- Normal – this is ideal for most use cases and is the default setting for all websites.
- Medium – this is ideal for websites that are integrated with other web services and require a bit more throughput.
- High – this is for high load traffic websites, including content management using the web file browser or REST API that might require additional memory or CPU.

Custom applications can only be configured on private websites. Custom application containers run in unprivileged mode and with a specific user ID and group ID. This means that users cannot configure Docker images where processes run with the root user, thereby decreasing the risk if using a compromised Docker image.

A set of *PriorityClasses* defines three priority levels: Normal, Medium, and High. These ensure that critical workloads receive the resources they need even in times of contention, maintaining the stability and reliability of essential websites and core components:

- Normal – by default, all websites get this priority class.
- Medium – websites that are monitored, usually public websites, get this priority instead of the low priority class.
- High – all the platform components, i.e.: Hub, Operator, Redis and all Jobs, get this priority class.

Websites can only be configured as public by *Hub* administrators, ensuring users need to go through an approval process. This decreases the risk of sensitive content exposure or non-compliant ECMWF design. Private websites can be configured with two-factor authentication, ensuring that users must enter a temporary one-time token (TOTP) before accessing the website, providing an extra level of security. Each website has a renewable retention date of one year from creation.

Ecosystem

ECMWF Sites offers a Command Line Interface (CLI) and a Software Development Kit (SDK).

These tools significantly enhance the platform ecosystem by providing users with powerful, flexible, and efficient ways to interact with the platform. They take full advantage of the *Hub* and the websites' REST APIs. Both the CLI and the SDK allow users to automate tasks, manage resources, and perform operations directly from the command line, streamlining workflows and improving productivity. Together, these tools empower users to fully leverage the platform's features, driving greater adoption and facilitating its usage.

Performance and load testing

To illustrate the performance of the system, a set of test scenarios was created. These tests consist in measuring the performance of a website in terms of content management, by uploading a set of static HTML and plot data of around 1.5 GB, and then accessing the website and navigating to a second location within that same website. The website is configured as Public and with Resource Limits set to High, in what we consider the ideal condition. These tests were executed from the three locations where ECMWF is based, Bologna (Italy), Bonn (Germany), and Reading (UK). They provide a good insight into the overall performance of the service across Europe.

The results of the test runs can be seen in Table 1 and Table 2, while Figure 6 and Figure 7 show the consumption of resources throughout the duration of these tests (circa 1 hour and 30 minutes per run). These results show that the website performs at the highest level under the tested load. In these performance tests, while the website is configured with High resources, CPU is not limited, hence leading to the highest throughput. Memory consumption is very low for these operations, due to a great optimisation of the REST API. Usually, high memory consumption comes from the use of either custom applications or the web file browser open-source application: *https://github.com/filebrowser/filebrowser*.

These and other performance tests can be found in detail at *https://sites.ecmwf.int/performance/results/*.

		Reference (Local)	Italy (Bologna)	Germany (Bonn)	UK (Reading)
50 Users	Apdex (Application Performance Index)	0.998	1,000	0.885	0.994
	90% (milliseconds)	84	99	892	237
	Throughput (requests/second)	73	74	69	76
25 Users	Apdex (Application Performance Index)	1,000	1,000	0.979	0.997
	90% (milliseconds)	32	77	275	177
	Throughput (requests/second)	76	83	57	64
10 Users	Apdex (Application Performance Index)	1,000	1,000	0.999	0.997
	90% (milliseconds)	13	73	96	157
	Throughput (requests/second)	65	51	40	32

Table 1 This table shows performance test results for the websites' content management under different load and different locations. Website content management performs very well, with 90% of the requests being served in 237 ms or less in the UK and in 892 ms or less in Germany. Content management is normally performed by a single user at a time, so these results show great REST API performance at scale.

		Reference (Local)	Italy (Bologna)	Germany (Bonn)	UK (Reading)
250 Users	Apdex (Application Performance Index)	1,000	1,000	0.999	1,000
	90% (milliseconds)	10	36	55	138
	Throughput (requests/second)	190	190	190	190
110 Users	Apdex (Application Performance Index)	1,000	1,000	0.999	1,000
	90% (milliseconds)	10	46	57	138
	Throughput (requests/second)	190	190	190	190
50 Users	Apdex (Application Performance Index)	1,000	1,000	0.999	1,000
	90% (milliseconds)	10	32	62	137
	Throughput (requests/second)	50	50	50	50

Table 2 This table shows the performance test results for website access under different load and different locations. Website access performs very well, with 90% of the requests being served in 138 ms or less in the UK and in 62 ms or less in Germany.

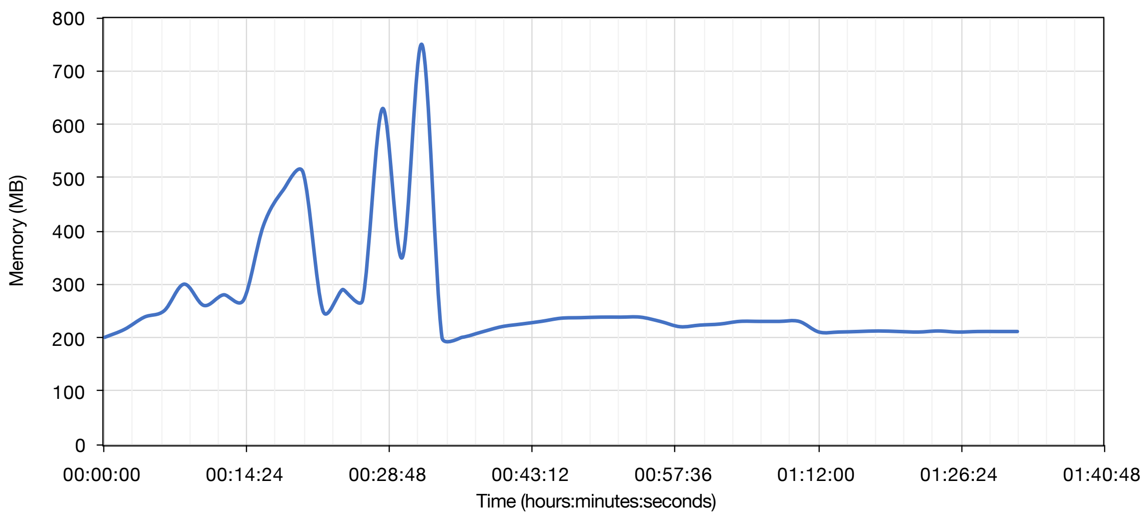


Figure 6 This figure shows the total memory usage throughout the performance tests. Memory peaks at 750 MB while performing content management, while during access to the website the peak reduces to only 238 MB.

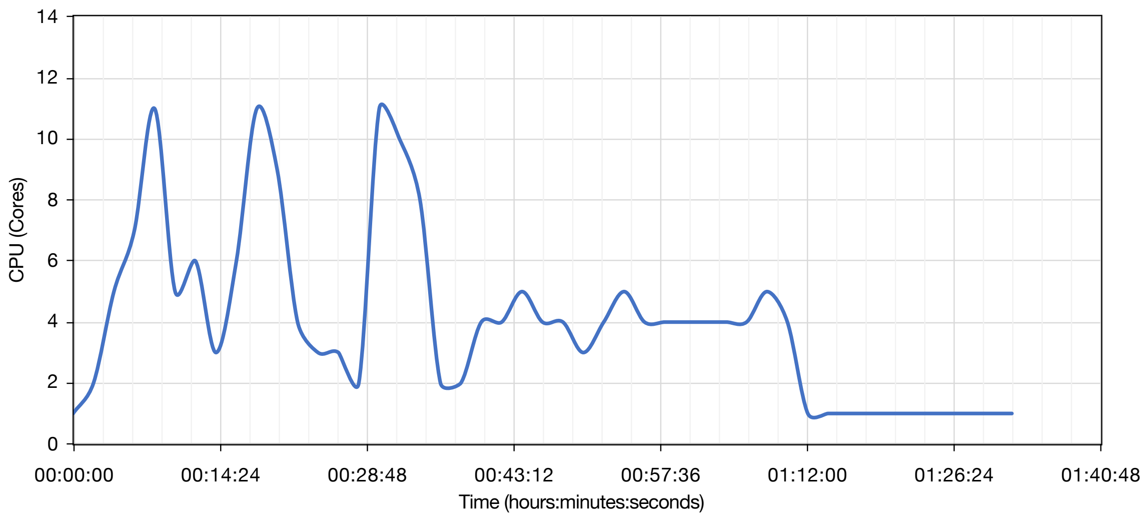


Figure 7 This figure shows the total CPU (Cores) usage throughout the performance tests. CPU usage peaks at 11 CPU Cores while performing content management, while during access to the website the peak reduces to 5 CPU Cores.

Concluding remarks

In the past months, several developments have been made to improve the way the main components of the platform are distributed. The idea behind these changes is to open-source the project, decoupling the main components from ECMWF's specifics, and to make it even more configurable. Such a platform offers a robust system that empowers users to create, manage, and customise online spaces tailored to a diverse range of use cases and can be highly advantageous for other organisations or national meteorological services.

Several improvements on the Site REST API application were made while migrating from Python to Golang. This led to big improvements in throughput, while significantly reducing the memory consumption compared to the previous implementation. The latest version of the Site REST API supports streaming of large files, enabling *ECMWF Sites* to be used for sharing large datasets over HTTP. Looking ahead, the continued development of *ECMWF Sites* will focus on meeting evolving user needs, strengthening security measures, and furthering its potential as an open-source solution, ensuring it remains a reliable and adaptable platform for diverse use cases.

It is important to note that *ECMWF Sites* was introduced during the preparation for our data centre migration in 2022, providing a straightforward way to host many services without the need for custom integrations or involvement from other teams. This alleviated some of the pressure on the web development team within ECMWF's Forecasts and Services Department.

Further reading

Martins M., 2020: Introducing Sites: static websites as a service. *ECMWF Newsletter* **No. 164**, 17. *https://www.ecmwf.int/en/newsletter/164/news/introducing-sites-static-websites-service*

Varela D. & **Martins M.**, 2018: Re-Architecture of the Atlassian Collaboration Tools. *ECMWF Newsletter* **No. 157**, 17. *https://www.ecmwf.int/en/newsletter/157/news/re-architecture-atlassian-collaboration-tools*

Kubernetes, 2024: Concepts, *https://kubernetes.io/docs/concepts/*, accessed on 20 August 2024.

Kubernetes, 2024: Operator Pattern, *https://kubernetes.io/docs/concepts/extend-kubernetes/operator/*, accessed on 20 August 2024.

Kubernetes, 2024: Custom Resources, *https://kubernetes.io/docs/concepts/extend-kubernetes/api-extension/custom-resources/*, accessed on 20 August 2024.

ECMWF Confluence, 2024: User Documentation, ECMWF Sites - Websites as a Service, *https://confluence.ecmwf.int/display/UDOC/ECMWF+Sites+-+Websites+as+a+Service*, accessed on 20 August 2024.

Apdex (Application Performance Index), 2024: The Apdex Users Group, *https://ww.apdex.org*, accessed on 12 September 2024.