The Astrophysical Journal Supplement Series, 266:33 (11pp), 2023 June https://doi.org/10.3847/1538-4365/acc938

GWCLOUD: A Searchable Repository for the Creation and Curation of Gravitational-wave Inference Results

A. Makai Baker[1,2], Paul D. Lasky[1,2], Eric Thrane[1,2], Gregory Ashton[3], Jesmigel Cantos[4], Lewis Lakerink[4], Asher Leslie[4], Gregory B. Poole[4], and Thomas Reichardt[4]
[1] School of Physics and Astronomy, Monash University, Clayton, VIC 3800, Australia; paul.lasky@monash.edu, eric.thrane@monash.edu
[2] OzGrav: The ARC Centre of Excellence for Gravitational Wave Discovery, Clayton, VIC 3800, Australia
[3] Department of Physics, Royal Holloway, University of London, TW20 0EX, UK
[4] Astronomy Data and Computing Services (ADACS), The Centre for Astrophysics & Supercomputing, Swinburne University of Technology, P.O. Box 218, Hawthorn, VIC 3122, Australia
Received 2022 April 27; revised 2023 February 20; accepted 2023 March 3; published 2023 June 7

Abstract

There are at present $\mathcal{O}(100)$ gravitational-wave candidates from compact binary mergers reported in the astronomical literature. As detector sensitivities are improved, the catalog will swell in size: first to $\mathcal{O}(1000)$ events in the A+ era and then to $\mathcal{O}(10^6)$ events in the era of third-generation observatories like Cosmic Explorer and the Einstein Telescope. Each event is analyzed using Bayesian inference to determine properties of the source including component masses, spins, tidal parameters, and the distance to the source. These inference products are the fodder for some of the most exciting gravitational-wave science, enabling us to measure the expansion of the universe with standard sirens, to characterize the neutron-star equation of state, and to unveil how and where gravitational-wave sources are assembled. In order to maximize the science from the coming deluge of detections, we introduce GWCLOUD, a searchable repository for the creation and curation of gravitational-wave inference products. It is designed with five pillars in mind: uniformity of results, reproducibility of results, stability of results, access to the astronomical community, and efficient use of computing resources. We describe how to use GWCLOUD with examples, which readers can replicate using the companion code to this paper. We describe our long-term vision for GWCLOUD.

Unified Astronomy Thesaurus concepts: Gravitational waves (678)

1. Introduction

The recent release of the third LIGO–Virgo–KAGRA (LVK) Gravitational-Wave Transient Catalog (GWTC-3) increased the total number of gravitational-wave events detected by LVK to 90 (Abbott et al. 2021a). Olsen et al. (2022) detect an additional 10 events with the so-called "IAS pipeline," while Nitz et al. (2023) detect seven events not included in GWTC-3. The GWTC-3 catalog consists mostly of binary black hole mergers, with two binary neutron-star mergers and $\gtrsim$ two neutron star + black hole mergers.[5] Each event is characterized by $\approx$15–17 astrophysical parameters (Veitch et al. 2015). There are seven extrinsic parameters describing the location and orientation of the event with respect to the observatory and eight or more intrinsic parameters, including the component masses and spin vectors. Tidal parameters are often included for systems with a neutron-star candidate (Lackey & Wade 2015; Abbott et al. 2018; Chatziioannou 2020), and some binary black hole analyses now include parameters characterizing orbital eccentricity (Lower et al. 2018; Romero-Shaw et al. 2019, 2020a; Lenon et al. 2020; Romero-Shaw et al. 2021; Gayathri et al. 2022).

Each event is analyzed with a Bayesian inference pipeline (Veitch et al. 2015; Lange et al. 2018; Ashton et al. 2019b; Biwer et al. 2019; Romero-Shaw et al. 2020b) in order to determine its astrophysical parameters. The outputs of these pipelines typically include posterior samples—discrete representations of the posterior distribution, which can be used to calculate credible intervals for different combinations of astrophysical parameters (corner plots). However, their usefulness does not end there, as they serve as reduced data products for some of the most exciting gravitational-wave science. Posterior samples are used for population studies (Thrane & Talbot 2019; Vitale et al. 2022) to probe how compact binaries are distributed in mass and spin, providing insights into stellar evolution and binary formation (see, e.g., Abbott et al. 2019, 2021b, 2023). They are used in standard siren analyses (Schutz 1986; Holz & Hughes 2005) to measure cosmological expansion (see, e.g., Abbott et al. 2023) and to determine the neutron-star equation of state (Lackey & Wade 2015; Abbott et al. 2018; Baiotti 2019; Landry & Essick 2019; Vivanco et al. 2020; Wysocki et al. 2020).

While the outputs of inference pipelines are crucially important to gravitational-wave astronomy, there are several aspects of these inference products that make life complicated for gravitational-wave astronomers. First, it is often necessary to carry out many different inference calculations, which analyze the same event with subtle differences. In particular, there are several popular waveform approximants used to model gravitational waveforms, each with different capabilities and different systematic errors. It is therefore common to analyze each event with multiple waveforms. Likewise, it is sometimes necessary to analyze the same event with different prior assumptions (for example, assuming one or both compact objects are not spinning; Galaudage et al. 2021), which can also

[5] The event GW190814 (Abbott et al. 2020a) could be a binary black hole or a neutron star + black hole binary.

lead to multiple runs. There may also be runs carried out with different samplers or different sampler settings, which occasionally yield qualitatively different results (see, e.g., Chia et al. 2022; Vajpeyi et al. 2023). Finally, the data themselves can have different versions, due to variations in calibration (Sun et al. 2020), cleaning (Abbott et al. 2021c), and/or glitch subtraction schemes (Chatziioannou et al. 2020).

The large number of inference results associated with each event creates a bookkeeping problem. This problem is compounded by a second issue: the high computational cost of gravitational-wave inference. Even a "fast" run on an ordinary binary black hole event with a cheap approximant can take $\approx$10 hr. More ambitious analyses on longer signals and/or with cutting-edge approximants can take weeks. Given the substantial costs in time and CO_2 emission associated with the generation of astrophysical inference results, it is becoming increasingly necessary to carefully curate gravitational-wave inference results.

Finally, the lack of a centralized repository for inference results makes the current workflows of gravitational-wave astronomers inefficient and susceptible to error. A researcher looking for the output of a particular inference run—e.g., using the IMRPHENOMXPHM approximant (Pratten et al. 2021) to analyze GW151226 (Abbott et al. 2016a) with special sampler settings—may need to email collaborators to find the results. The results in question may have been subsequently moved or even deleted. And the researcher cannot be certain that the files she tracks down are precisely what she is looking for.

The situation is already challenging with 90 events. The difficulties will, of course, increase as the gravitational-wave catalog swells (Baibhav et al. 2019) to $\mathcal{O}(1000)$ events in the A + era and $\mathcal{O}(10^6)$ events in the era of third-generation observatories, such as Cosmic Explorer (Reitze et al. 2019; Evans et al. 2021) and the Einstein Telescope (Maggiore et al. 2020). In order to address these challenges, we introduce GWCLOUD, a searchable repository for the creation and curation of gravitational-wave inference results. There are five pillars underpinning its design philosophy:[6]

1. *Uniformity of results.* Inference results are downloaded and uploaded in a uniform format. Uniformity facilitates validation: new results must pass checks to ensure that the inference output is complete and uncorrupted, with the necessary metadata to repeat the analysis.
2. *Reproducibility of results.* By curating the metadata and code version of each result, we insist that every entry in GWCLOUD can be reproduced.
3. *Stability of results over time.* Each result is assigned a permanent location. Users can locate previous results using a search engine. Before launching a new inference job, users can search to see if the analysis they want has already been performed. Avoiding duplicate analyses reduces the carbon footprint of gravitational-wave astronomy.
4. *Access to results.* While a large fraction of gravitational-wave astronomy effort takes place within the LVK collaboration, significant advances are now made by external groups. GWCLOUD provides multiple levels of access so that results can be shared both within the LVK collaboration and to the larger astronomical community, facilitating the exchange of ideas among a broad community.
5. *Efficient use of computing resources.* GWCLOUD enables users to submit inference jobs on multiple computing clusters through a single portal. Each cluster can use different batch queuing protocols (e.g., SLURM versus CONDOR) and allow for different user groups (e.g., LVK users versus the general public). In this way, GWCLOUD helps match users with computing resources.

GWCLOUD is not the only tool that has been created to tackle these challenges. The Gravitational Wave Open Science Center (GWOSC) provides access to most of the publicly available posterior samples used in LVK papers. The samples can be queried, discovered, and downloaded through the GWOSC Event Portal, at https://gwosc.org/eventapi. These samples are also available through https://zenodo.org. However, the GWOSC database only contains results associated with LVK papers. It does not include inference results from "short-author" work that reanalyzes LVK events, e.g., with different waveforms. Nor does it include results for events not claimed as detections by LVK (Olsen et al. 2022; Nitz et al. 2023). GWCLOUD aims to provide a single repository where all inference results can be downloaded and where new jobs can be launched.

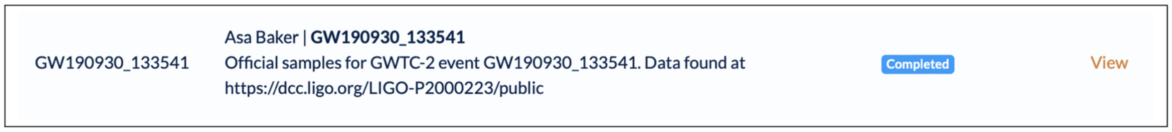
Recently, Williams et al. (2023) introduced ASIMOV, a framework for coordinating parameter estimation workflows. It includes a number of useful features, including a review sign-off system so that key results are vetted by humans. Meanwhile, the program PESUMMARY (Hoy & Raymond 2021) has helped facilitate the dissemination of uniform results (while simultaneously providing a tool for the visualization of inference results). It includes functionality to access result files[7] (both public and private) and functionality to reproduce results.[8] PESUMMARY, ASIMOV, and GWCLOUD provide complementary services, although the way in which they will interact in the future is not yet clear. In our vision, ASIMOV generates official LVK inference results for each event, which are then uploaded to GWCLOUD, where they can be searched for alongside a database that includes LVK auxiliary results (e.g., to see how the results of some event change assuming a different prior) and non-LVK results performed for short-author studies. It may be possible for GWCLOUD to at some point call PESUMMARY in order to visualize inference results.

The remainder of this paper is organized as follows. In Section 2, we cover the basics of GWCLOUD: how to submit new inference jobs and how to upload the results of an inference analysis. In Section 3, we provide the first of three case studies: we use GWCLOUD to reanalyze the iconic event GW150914 (Abbott et al. 2016b), but with the assumption that both black holes have negligible spin. In Section 4, we present the second case study: using GWCLOUD to investigate correlations between mass and spin parameters using events in the second gravitational-wave transient catalog (GWTC-2; Abbott et al. 2021d). In Section 5, we describe the third case study: using GWCLOUD to download posterior samples for the remarkably high-mass event GW190521 (Abbott et al. 2020b) obtained using an eccentric waveform approximant. We

[6] These design pillars are aligned with the Australian Research Data Commons guidelines for "FAIR" data, which are findable, accessible, interoperable, and reusable. For more information, see https://ardc.edu.au/resources/aboutdata/fair-data/.

[7] https://lscsoft.docs.ligo.org/pesummary/stable_docs/gw/fetch.html

[8] https://lscsoft.docs.ligo.org/pesummary/stable_docs/gw/cli/summaryrecreate.html


GW190930_133541

Asa Baker | **GW190930_133541**
Official samples for GWTC-2 event GW190930_133541. Data found at https://dcc.ligo.org/LIGO-P2000223/public

Figure 1. Example entry in GWcloud.

conclude in Section 6 with a discussion of future development plans. Technical details are provided in the Appendix. The case studies presented in this paper are supported by Jupyter notebooks available as part of the online supplement in GitHub (https://git.ligo.org/gwcloud/paper/) and Zenodo at doi:10.5281/zenodo.7783583.

2. Basics

2.1. What is GWcloud?

In this section, we provide a high-level overview of GWcloud and instructions for the most basic tasks users are likely to perform. GWcloud consists of two components: a portal to launch inference jobs and a database to store the results of inference jobs. Both components can be accessed using a web-based graphical user interface (UI) at https://gwcloud.org.au/. Users who prefer to access GWcloud entirely with command-line programming may instead use the application programming interface (API). We anticipate that the UI's job submission feature will be most useful for new and casual users. However, the UI's search feature should be useful to any user searching for old inference results. The API is likely to be most useful to experts who sometimes need to submit large batches of jobs. It allows for more complicated job submissions with features that are not supported using the UI (for example, custom priors). See Figures 6–7 of the Appendix for a description of the frontend and backend architecture of GWcloud. For a visualization of the principles underpinning the development of the UI and API, see Figure 8 of the Appendix.

The portal launches inference jobs using Bilby (Ashton et al. 2019b; Romero-Shaw et al. 2020b). Jobs launched through the UI are at present run on the OzStar cluster based at Swinburne University. Jobs submitted by authenticated LVK users through the API can also be run on computers that form part of the LIGO Data Grid. It is also possible to upload jobs to the GWcloud database that were not run through the GWcloud portal, so long as they are in the standard Bilby format (see Section 2.3).[9] This is useful for storing jobs that were run before the creation of GWcloud or jobs that require special resources to run, for example, computationally expensive parallel Bilby (Smith et al. 2020) analyses that require a high-performance computing cluster.

Users visiting the GWcloud landing page are met with a prompt requiring sign-in. Members of the LVK collaboration can sign in using their `albert.einstein` credentials, which also provides access to the LIGO Data Grid, while other users can create a GWcloud account. After logging in, the user is taken to the "public jobs" page, which lists the most recent GWcloud runs; see Figure 1 for an example entry from this recent job list. The search field allows users to find jobs based on their `description`, the `user` who submitted the job, the `job_name`, and the `event ID`. `Labels` are available to distinguish some jobs as special. For example, the `Official` label indicates that a job is used for an official LVK result. Other currently available labels include `Bad run`, `Production run`, `Review requested`, and `Reviewed`.[10] Previous jobs can be viewed and downloaded by clicking on the appropriate `view` link. Users may create a new job by clicking on `start a new job` and following the instructions.

In the next subsections, we describe how to submit (or upload) a job using the API. Additional information is provided on the GWcloud web page by clicking on `Python API`. In order to implement the examples below, readers must install the GWcloud API:

```
pip install gwcloud − python.
```

2.2. Submitting a New Job with the GWcloud API

Here, we describe a Python script for submitting a new GWcloud job using the API; see `JobSubmission.ipynb` for the corresponding Jupyter notebook. The corresponding job can be viewed on the GWcloud UI by searching for the name: `GW150914Example`.

The first step is for the user to authenticate by initializing a token identifier generated by GWcloud. At the beginning of any GWcloud script, include the following lines to import the GWcloud API and set up your token:

```
from gwcloud_python import GWCloud, Cluster
gwc = GWCloud(token = ′YourTokenHere′).
```

The next step is to create a Bilby `.ini` file, which is required to submit a job with GWcloud because it is required to run Bilby. The `.ini` file for this tutorial is `GW150914_example.ini`. The `.ini` file tells Bilby which data to analyze and how to analyze them.[11] The `.ini` file contains local paths to noise power spectral density (`PSD`) file(s), spline calibration (`.calib`) file(s), and the `.prior` file. All of these files are uploaded to GWcloud for reproducibility. With the `.ini` file ready, we submit the job to the LIGO Data Grid's Caltech cluster like so:

```
new_job = gwc.start_bilby_job_from_file(
job_name = ″GW150914Example″,
job_description = ″TestingGWCloud″,
private = False,
ini_file = ″GW150914_example.ini″,
cluster = Cluster.CIT).
```

[9] This feature can be used to upload results from other inference codes, such as LALInference (Veitch et al. 2015) or RIFT (Lange et al. 2018).

[10] Labels like `Official` and `Bad Run` are intended to provide users with information on whether some jobs are trustworthy or not. However, buyer beware: jobs bearing no label have not necessarily been inspected for quality control.

[11] Please see https://lscsoft.docs.ligo.org/bilby/ for additional Bilby documentation.

Once this command is executed, a new job with `job_name = GW150914Example` becomes visible on the GWCloud UI.[12] Since we set `private = False`, the job can be viewed by anyone using GWCloud.[13] The last line of code tells GWCloud to run this job on the Caltech computing cluster.

The progress of the new job can be monitored with the GWCloud UI. When the job is complete, the API can be used to retrieve the posterior samples from GWCloud with the following command,

```
new_job.save_result_json_files('/path/'),
```

which saves the result files containing the posteriors to the specified path. For an overview of the interaction between the submitted job and the remote computing cluster, see Figure 9 of the Appendix.

2.3. Uploading the Results of an Existing Bilby Run

Here, we describe a Python script to upload existing results to GWCloud job using the API; see `JobUpload.ipynb` for the corresponding Jupyter notebook. The corresponding job can be viewed on GWCloud by searching for `job_name = GW150914` by `user = Asa Baker`. As our starting point, we need a Bilby `output/` directory with the requisite subdirectory structure. We modify the `label` field in the `*_config_complete.ini` file to set the GWCloud job name, e.g.,

```
label = 'GW150914_Upload_Example'.
```

Next, create a tar-zipped file of the Bilby `output/` directory, which can be accomplished by running this command:

```
tar − cvf archive.tar.gz.
```

Finally, the job is submitted by uploading the tar-zipped file to GWCloud:

```
gwc.upload_job_archive("Exampleupload
with GW159014.",
"/path/archive.tar.gz").
```

GWCloud checks the submission to make sure all the requisite results and supporting files are included.

3. Case Study I: Submit a Job to Analyze GW150914 with a Zero-spin Prior

The GWCloud graphical UI allows users to submit inference jobs with various default prior settings. While these settings are probably adequate for new users, expert users will need the API in order to perform runs with custom priors. Here, we provide an example of how the API can be used to carry out an inference calculation with a nonstandard prior; see `Case-Study1.ipynb` for the corresponding Jupyter notebook. The corresponding jobs can be viewed on GWCloud by searching for the names: `GW150914Example` and `GW150914NoSpin` by `Asa Baker`. Specifically, we reanalyze the iconic first binary black hole event GW150914 (Abbott et al. 2016b), but assuming that both black holes have negligible dimensionless spins $\chi_1 = \chi_2 = 0$ (here, the "1" subscript refers to the more massive "primary" black hole, while the "2" subscript refers to the less massive "secondary" black hole). This example is motivated by works by Fuller & Ma (2019), Miller et al. (2020), Roulet et al. (2020), Galaudage et al. (2021), and Hoy et al. (2022), which suggest that a subpopulation of LVK detections are likely characterized by negligible black hole spin.

We prepare two `.ini` files: `GW150914_example.ini` reproduces standard Bilby settings for a short-duration (high-mass) binary black hole signal. The prior for the dimensionless spins χ_1, χ_2 is uniform on the interval of zero to one. Meanwhile, in `GW150914_nospin.ini`, we set $\chi_1 = \chi_2 = 0$. We submit both jobs and download the results using the syntax described in Section 2.2. In Figure 2, we provide a corner plot comparing the credible intervals for various parameters of GW150914, assuming a uniform prior for the dimensionless spins (blue) and a no-spin prior (orange). The different shadings indicate 1σ, 2σ, and 3σ credible intervals. The different choice of prior yields subtle but interesting shifts in the posterior distribution. Comparing the marginal likelihoods for each run, we find that the zero-spin hypothesis is preferred with a Bayes factor of $\mathrm{BF} = 3.517$, consistent with the conclusions from Miller et al. (2020), Roulet et al. (2020), Galaudage et al. (2021), and Hoy et al. (2022) that some gravitational-wave events are best described as having negligible spin.

4. Case Study II: Download Results from GWTC-2 for Correlation Study

In this case study, we provide an example of how the API can be used to download previous inference results to look for trends in the population of merging binary black holes; see `CaseStudy2.ipynb` for the corresponding Jupyter notebook. The corresponding jobs can be viewed on GWCloud by searching for the keyword: `GWTC-2`. This example is motivated by work by Callister et al. (2021), suggesting that black hole spin is correlated with mass ratio. We download the "preferred samples" (used for official LVK analyses) for 47 binary black hole events in GWTC-2 (Abbott et al. 2021d, 2023). To retrieve these GWTC-2 jobs, we run the following command:

```
jobs = gwc.get_public_job_list(
search = "GWTC − 2",
time_range = TimeRange.ANY).
```

In Figure 3, we plot the 90% credible intervals in the plane of total mass M and mass ratio q for events in GWTC-2. In Figure 4, meanwhile, we plot credible intervals in the plane of chirp mass $\mathcal{M}$ and the effective inspiral spin χ_{eff}. These two plots can be compared to Figures 6–7 in Abbott et al. (2021d). By examining the distributions of events in two-dimensional planes, it is sometimes possible to see previously unknown correlations. As is shown in Abbott et al. (2021d), in this case, there is not an obvious correlation present in either plot.

[12] The job name, combined with the GWCloud user name of the person who submitted the job, uniquely define each event. Thus, two different users can have a job named `GW150914Example`, but one user cannot give this name to two different jobs.

[13] Jobs are marked as `LVK` or not. They are also marked as `private` or not. A job with `LVK = true` and `private=false` may be viewed by all members of the LVK Collaboration.

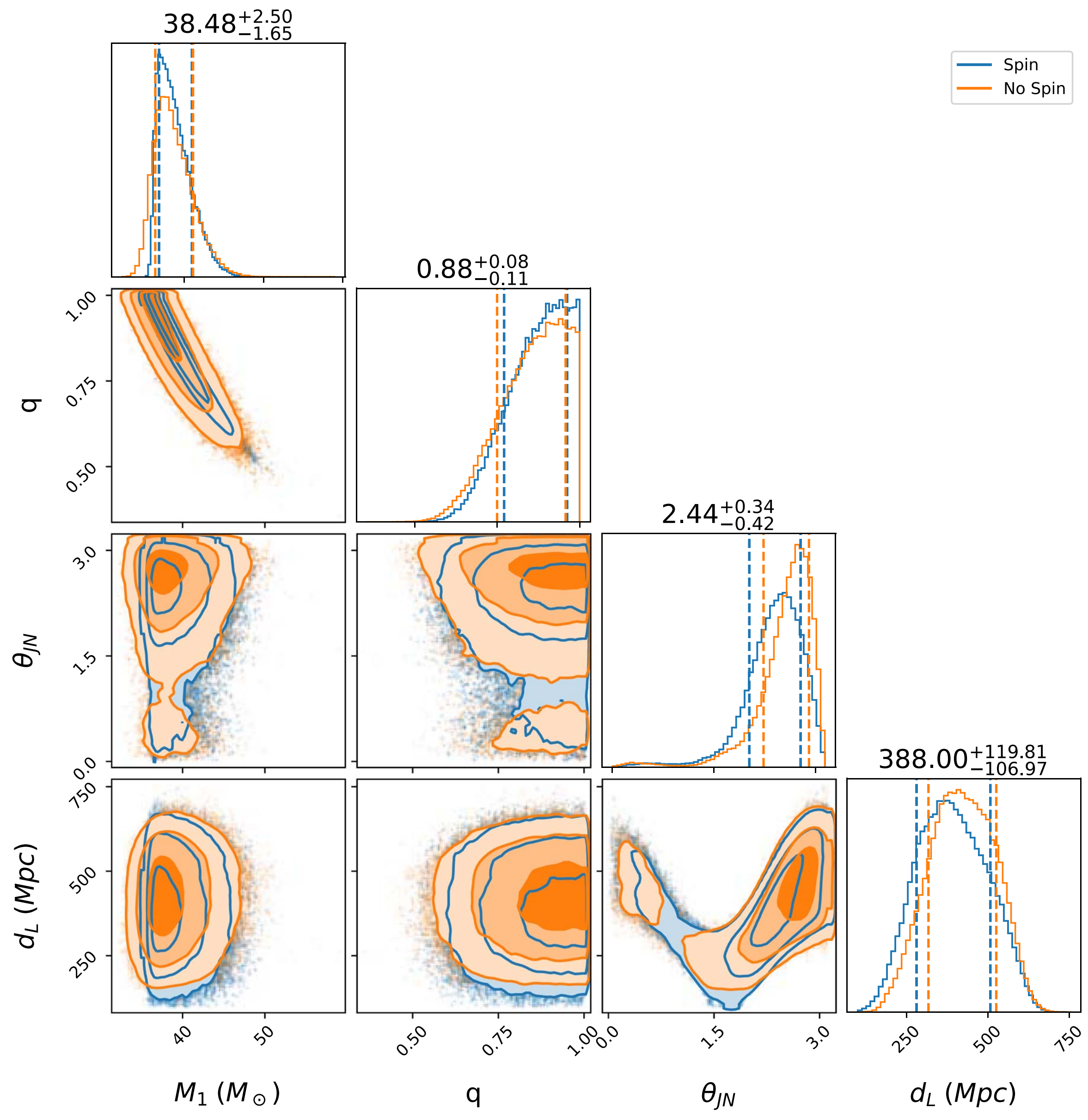


Figure 2. A corner plot showing the marginalized posterior distribution of the first binary black hole event GW150914. The masses are provided in the lab frame. The default results—calculated with a $U(0, 1)$ prior for the dimensionless spins χ_1, χ_2—are shown in blue, while the orange shows the results assuming $\chi_1 = \chi_2 = 0$. The different shades indicate 1σ, 2σ, and 3σ credible intervals.

5. Case Study III: Download Results for Eccentric Analysis of GW190521

Binary black holes formed from stellar binaries are expected to merge with quasi-circular orbits. However, a nonzero eccentricity may indicate that the binary was assembled from previously unbound black holes, a process called "dynamical formation." We consider GW190521 (Abbott et al. 2020b), one of the most massive binary black hole events to date, which shows signs of nonzero spin precession and/or eccentricity (Romero-Shaw et al. 2020a; Gayathri et al. 2022). Romero-Shaw et al. (2020a) analyzed this event using quasi-circular and eccentric waveforms. The results of this analysis have been uploaded to GWCLOUD and can be viewed by searching for "GW190521 Asa Baker" (the job name and the user, separated by spaces), which will turn up at least two entries. One entry, `job_name = GW190521, user = Asa Baker`, contains results obtained with the circular waveform obtained with NRSUR7DQ4 (Varma et al. 2019). Another entry, `job_name = GW190521_eccentric, user = Asa Baker`, contains results obtained with SEOBNRE (Cao & Han 2017; Liu et al. 2020).

To retrieve these jobs from GWCLOUD, search for `GW190521` in the public jobs by executing the following command,

```
jobs = gwc.get_public_job_list(
search = "GW190521",
time_range = TimeRange.ANY),
```

and download the jobs by `Asa Baker`. In Figure 5, we plot the posterior distribution for the eccentricity of GW190521 at a reference frequency of 10 Hz (compare with Figure 1 of Romero-Shaw et al. 2020a). See `CaseStudy3.ipynb` for the corresponding JUPYTER notebook.

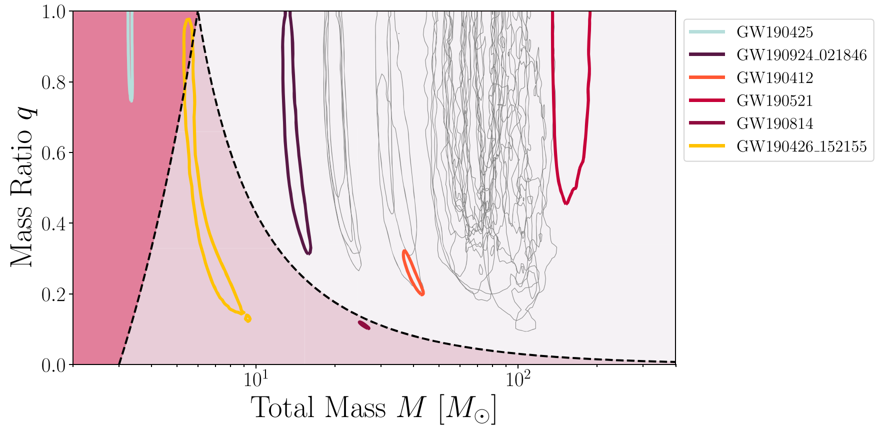


Figure 3. Compact binary coalescence events from GWTC-2 in the plane of total mass M and mass ratio q. Each contour represents the 90% credible region. Select events are highlighted. The dashed lines mark the border beyond which one or more component has a mass $<3\,M_\odot$; objects below this threshold are neutron-star candidates. The events in the gray region are confidently binary black hole events, while the events in the mauve region may contain a neutron star. The purple region is forbidden by the requirement that $m_1 > m_2$.

6. Future Development

We close by considering the future of GWCLOUD, describing the new functionality we hope to add in both the short term and long term. As we plan for the future, we invite input from the astronomical community; please visit our git issue tracker to leave a suggestion or to propose a new feature.[14]

6.1. Short-term goals

1. *Making LVK jobs public.* When LVK data are published, jobs that have previously been marked as `LVK` can be changed to `public`.
2. *GWCLOUD teams.* Share jobs among a small team. Team members can add comments to different jobs, e.g., "This result does not look fully converged." Teams can combine jobs to create catalogs.
3. *Archiving complementary information.* Gravitational-wave inference results do not exist in vacuums. In order to generate and interpret them, we rely on a number of other data products, including estimates of the noise power spectral density (e.g., Littenberg & Cornish 2015), injection studies used to quantify selection effects (Gerosa et al. 2020; Talbot & Thrane 2022), and probabilities that a given event is astrophysical $p_{\rm astro}$ (Kapadia et al. 2020). We hope to extend GWCLOUD to include these and other data products.
4. *Visualization.* Static and dynamic visualizations of inference products are useful to understand covariances. Such functionality is currently offered within the `pe_summary` toolkit (Hoy & Raymond 2021); a short-term goal is full integration of these visualization toolkits into the GWCLOUD workflow.
5. *Default priors.* A default prior is recommended for new jobs submitted using the UI if the job is associated with a known event.

6.2. Long-term goals

1. *Identify similar jobs.* Warn users if they are about to launch a job that is similar to one already in the database. Users may choose to use existing results rather than waiting for new ones (and potentially generating more CO_2 emissions). In some cases, importance sampling can be used to re-weight posterior samples to convert the results from a "proposal" distribution to a "target" distribution (Payne et al. 2019).
2. *Estimate job runtime.* Use machine learning to provide the estimated time to completion for new jobs. Warn users if they launch a job that is likely to take more than a week to complete.
3. *Connecting to other clusters.* Currently, GWCLOUD provides users with access to the computing clusters of the LIGO Data Grid and the OzStar clusters at the Swinburne University of Technology. However, GWCLOUD could be connected to other computing resources such as the Open Science Grid (Pordes et al. 2007).
4. *Automated inference.* The project could be extended to launch automated inference jobs for promising triggers by, e.g., integrating with ASIMOV (Williams et al. 2023). When extra computational resources are available, inference can be carried out on all data segments. The results can be used to carry out a statistically optimal search for the astrophysical background (Smith & Thrane 2018) and to construct fully Bayesian detection statistics (Veitch & Vecchio 2010; Ashton et al. 2019a; Pratten & Vecchio 2021).
5. *Beyond posterior samples.* The majority of gravitational-wave inference relies on posterior samples. However, in some cases, it can be useful to work with other inference products, for example, machine-learning (and grid) representations of marginal likelihoods (Lange et al. 2018; Vivanco et al. 2019; Vivanco et al. 2020; Wysocki et al. 2020). Additional work is required to define a standardized format for such inference products.

This work is supported by the Gravitational Wave Data Centre, which is funded under the Astronomy National Collaborative Research Infrastructure Strategy (NCRIS) Program via Astronomy Australia Ltd. (AAL). This work is supported by the Australian Research Council (ARC) Centre of Excellence CE170100004. P.D.L. is supported by ARC Discovery Project DP22010161. E.T. and P.D.L. are supported by ARC Discovery Project DP230103088. We would like to thank the referee for the constructive feedback on our manuscript.

[14] https://gitlab.com/CAS-eResearch/GWDC/projects/gwcloud/issues

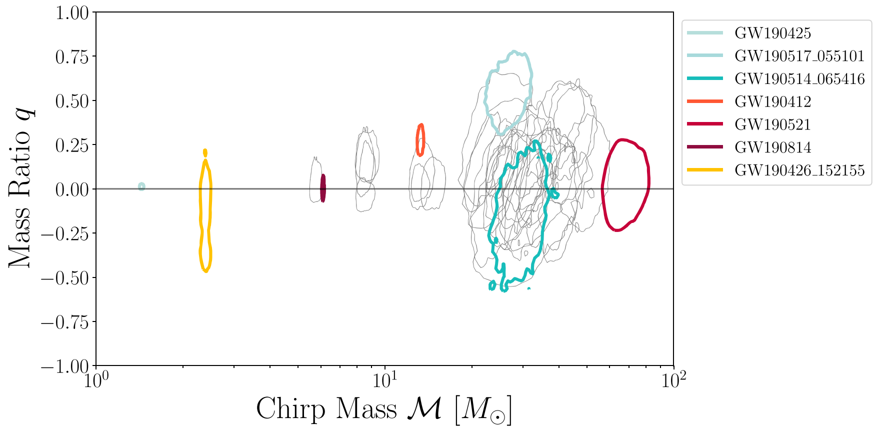


Figure 4. Compact binary coalescence events of the LVK GWTC-2 catalog in the plane of chirp mass $\mathcal{M}$ and effective inspiral spin $\chi_{\rm eff}$. Each contour represents the 90% credible region for a different event. Select events are highlighted.

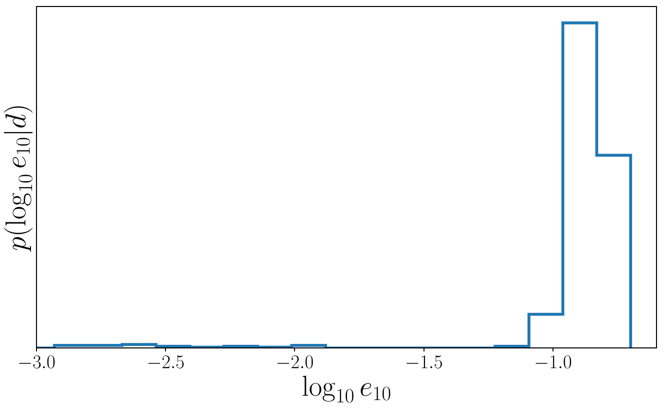


Figure 5. The posterior distribution for the eccentricity of GW190521 at a reference frequency of 10 Hz obtained with the SEOBNRE waveform (Cao & Han 2017; Liu et al. 2020) by Romero-Shaw et al. (2020a).

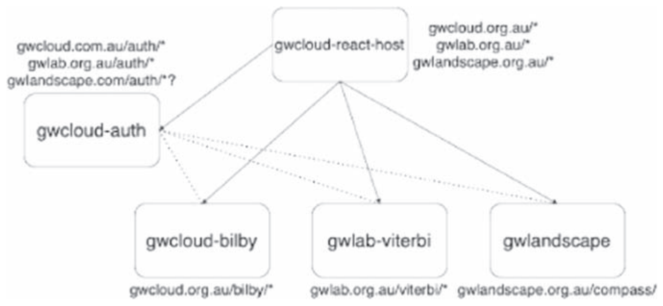


Figure 6. GWDC frontend architecture. This diagram shows how the frontend uses a microservice architecture depending on the URL being visited. The `React` host is always loaded, and is then responsible for loading `Auth` or application `javascript` bundles. This architecture prevents having a single monolithic `javascript` bundle that becomes difficult to maintain.

Appendix
Technical Details

GWCLOUD leverages a variety of modern web technologies to provide seamless access via web browsers or an API, exposed to researchers via terminal command line by a publicly available PYTHON client called `gwcloud-python`.[15] In this appendix, we provide additional details about GWCloud along with several graphical representations of various aspects of GWCloud. In Figure 6, we include a model of the frontend architecture. In Figure 7, we include a model of the GWDC service architecture. In Figure 8, we include an illustration of the deep-thinking process. In Figure 9, we include a high-level infrastructure architexture diagram.

A.1. Application Architecture

In the backend, GWCLOUD takes advantage of `Django`[16]: a mature PYTHON-based Model View Controller[17] web framework widely used in the commercial sector. Alongside `Django`, the chosen technology for API transport is `GraphQL`,[18] which provides an efficient and effective way for clients (such as `gwcloud-python` or any web browser) to request the data they require and only the data they require. This is in contrast to other less efficient API transport architectures, such as Representational State Transfer, which can lead to a variety of issues (e.g., request cascades[19]) when dealing with complex data.

In the frontend, the chosen technology is `React.js`,[20] which is an industry standard web framework that efficiently handles Document Object Model updates to generate fully interactive and dynamic web applications. To complement `React.js`, `Relay.js`[21] (which uses `GraphQL` fragments and caching to efficiently maintain a consistent state for web applications) makes the representation of the data from the web server more efficient.

GWCLOUD exists as one of several projects core to the Gravitational Wave Data Centre (GWDC):[22] a software engineering initiative of Astronomy Australia Limited,[23] based at Swinburne University of Technology and operated alongside the Astronomy Data and Computing Services (ADACS) team. ADACS is tasked with providing software services to the Australian astronomy community and the combined resources of it and the GWDC presently consist of approximately 15 dedicated software development professionals.

[15] See https://pypi.org/project/gwcloud-python/.
[16] https://www.djangoproject.com/
[17] https://developer.mozilla.org/en-US/docs/Glossary/MVC
[18] https://graphql.org/
[19] https://leapgraph.com/what-graphql-solves/
[20] https://reactjs.org/
[21] https://relay.dev/
[22] https://gwdc.org.au
[23] https://astronomyaustralia.org.au

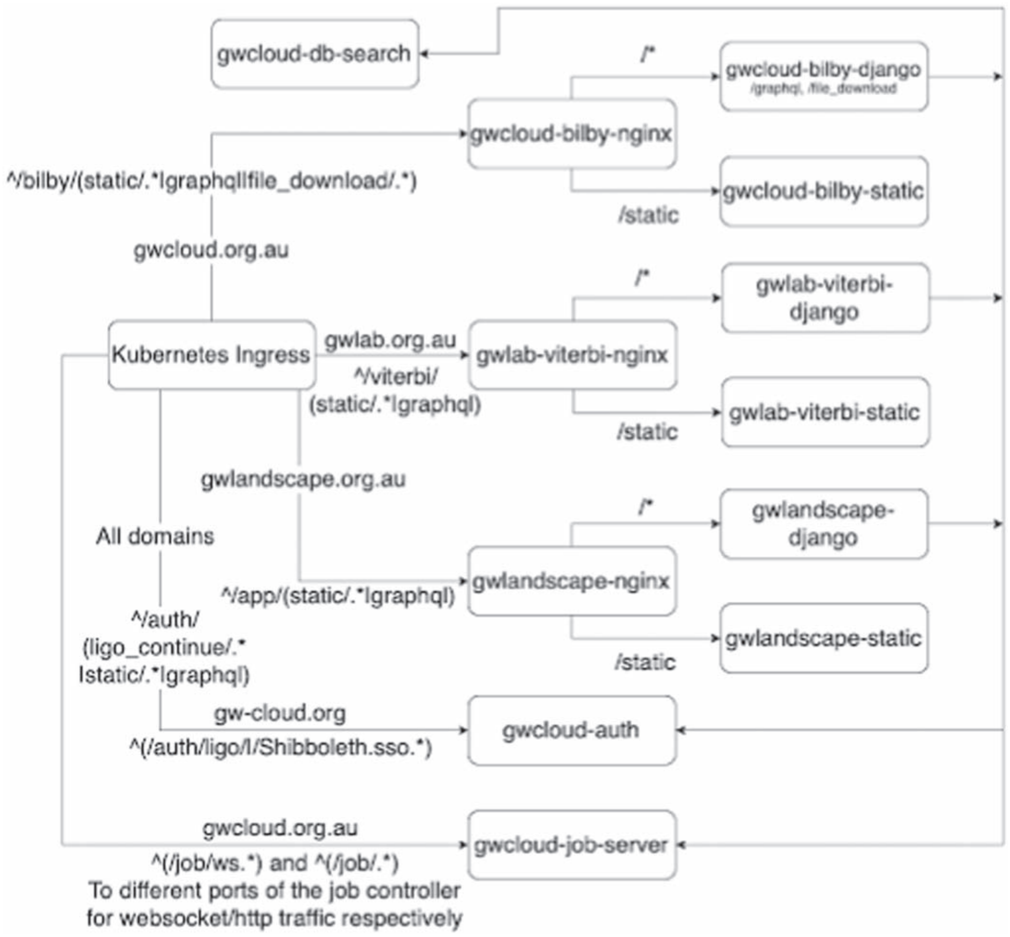


Figure 7. GWDC services architecture; virtual arrangement of backend services. All external requests originate from the `Kubernetes` ingress service and are distributed to the relevant internal services. Internal services communicate with other internal services, including the Authentication, Job Server, and Database Search services.

Several of the core GWDC projects are web applications (others include GWLAB[24] and GWLANDSCAPE[25]) with similar infrastructure requirements (e.g., LVK user authentication, execution of "jobs" on compute clusters hosting LIGO data, management, visualization, and searching of these jobs, etc.). To easily expand and grow these projects and to reduce maintenance overheads,[26] a microservice architecture was chosen for all GWDC web applications. When a new feature is to be added or a bug is found in an existing service, it is easy to identify and isolate the code involved, reducing complexity and technical debt. Such isolation also naturally simplifies testing, promoting enhanced reliability and uptime.

Microservice architectures consist of multiple discrete applications running behind the scenes to perform independent and unrelated tasks. The GWCLOUD application, for example, is just a single backend service and frontend `Javascript` bundle tasked purely with performing tasks related to the submission and management of Bilby jobs. Other notable services operating in parallel include an authentication service, a database search service, and a Job Controller. The authentication service facilitates integration with the LIGO Identity Provider[27] and manages the accounts of non-LIGO-affiliated users.[28] This service also provides details about the user (e.g., LVK membership status and user details such as name and email address, etc.). The database search service[29] provides efficient and powerful job searching based on the provided search parameters. Finally, the Job Controller provides a service that GWCLOUD and other projects can use to submit and monitor jobs on high-performance computing (HPC) facilities as well as to fetch the results and files of those jobs while running or once completed.[30]

GWCLOUD also loosely makes use of microservices in the frontend. The `React` host is responsible for loading services as required, depending on the current URL of the web browser. This takes advantage of `Webpack`[31] Federated Modules.[32]

A limited amount of shared code is present, including the host `React` module, which is responsible for orchestrating which project to load, depending on the currently visited URL. Other shared code exists between projects, although it is not shared from one location, but rather duplicated from a base

24 https://gwlab.org.au/
25 https://gwlandscape.org.au/
26 https://microservices.io/
27 https://ldvw.ligo.caltech.edu/

28 https://gwcloud.org.au/auth/
29 https://github.com/gravitationalwavedc/gwcloud_db_search
30 https://github.com/gravitationalwavedc/gwcloud_job_server
31 https://webpack.js.org/
32 https://webpack.js.org/concepts/module-federation/

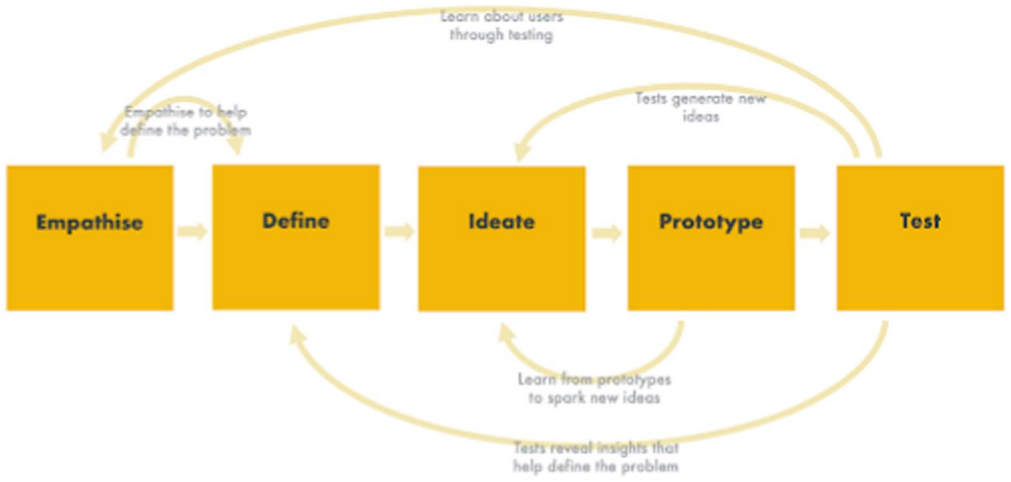

Learn about users through testing
Empathise to help define the problem
Tests generate new ideas
Empathise
Define
Ideate
Prototype
Test
Learn from prototypes to spark new ideas
Tests reveal insights that help define the problem

Figure 8. Design Thinking process. A flowchart showing the steps used by GWDC to develop UIs and APIs. This can be a sequential process of empathizing with the users, defining the issues they face, ideating solutions, creating a prototype, and then testing to determine if the prototype is successful. Often the insights and data discovered can better inform previous steps, leading to an iterative workflow.

project template that contains the aforementioned basic core functionality.

A.2. User Experience and Design

The GWDC implements a human-centered design (Norman 2013) approach in the creation of UIs and client-facing APIs. In most cases, the design thinking (Brown 2008) variant of human-centered design is used. The main goal of design efforts is to reduce the cognitive load of programming-related tasks to allow researchers to focus on scientific challenges. To our knowledge, this is the first time that human-centered design has been intentionally used to improve the UIs and APIs of gravitational-wave research applications.

GWCLOUD has undergone several usability tests (Nielsen 2000; Norman 2013) to inform and validate design choices for the UI and API. Initial usability tests were performed on the UI to develop an understanding of how researchers used the interface and ultimately to build empathy with their needs. These data were analyzed to define the problems researchers faced and to prototype design solutions. Once a solution had been selected, it was implemented and then validated with further testing. This iterative design process is ongoing, but has already seen improvement in reducing the number of errors, lowering the barrier of entry, and increasing the user satisfaction, efficiency, and learnability of the user interface (UI) and application programming interface (API).

A.3. Underlying Infrastructure

To reduce maintenance complexity, `Kubernetes` is used as the underlying infrastructure for GWCLOUD applications. `Kubernetes` is an open-source platform designed for containerized applications. It enables automated operations, such as deployments, backups, rollbacks, horizontal virtual resource scaling, name-spaced role-based access control, and configuration decoupling from applications.[33] Since virtual hosts are abstracted from deployments, applications can be redeployed as needed in an automated self-healing manner.[34]

Within `Kubernetes`, supporting tools are deployed in compliance with the cloud-native roadmap.[35] As per the roadmap, all applications involved with GWCLOUD are packaged and deployed in the form of Docker containers. Docker is an Open Container Initiative (OCI)-compliant[36] containerization platform enabling the ingestion of container configurations by other OCI-compliant tools, such as PODMAN or BUILDAH. The container images are stored in a container registry. These container images are then repackaged with default deployment configurations in the form of Helm charts and are stored to a Helm chart repository. Both container images and Helm charts are stored within SONATYPE NEXUS.[37] As a prerequisite, all sensitive data are declared and initialized within the centralized secrets manager Hashicorp Vault, through its key value pair secrets engine.[38] Values required by deployments from Hashicorp Vault must meet the access requirements configured within Vault.[39] The Helm charts are then ingested and deployed to the target `Kubernetes` cluster through ARGOCD:[40] the management tool for the deployment lifecycle of GWDC applications.

A.4. Interfacing with HPC Facilities

An underlying component of the GWDC infrastructure underpinning the success of several projects including GWCLOUD is the Job Controller: a module responsible for communicating with remote HPC resources such as those based at Swinburne University[41] and Caltech.[42] Previously, the only similar software solutions have generally been tightly coupled to single clusters, often requiring the cluster's filesystem to be mounted in a manner allowing web applications to access files directly. These solutions scale poorly and are hostile to many

[33] https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/
[34] https://kubernetes.io/docs/concepts/architecture/cloud-controller/
[35] https://github.com/cncf/trailmap/blob/master/CNCF_TrailMap_latest.pdf
[36] https://docs.docker.com/buildx/working-with-buildx/
[37] https://help.sonatype.com/repomanager3/nexus-repository-administration/formats/docker-registry
[38] https://www.vaultproject.io/docs/secrets/kv
[39] https://www.vaultproject.io/docs/platform/k8s/injector
[40] https://argo-cd.readthedocs.io/en/stable/operator-manual/architecture/
[41] https://supercomputing.swin.edu.au/
[42] https://computing.docs.ligo.org/lscdatagridweb/resources/index.html

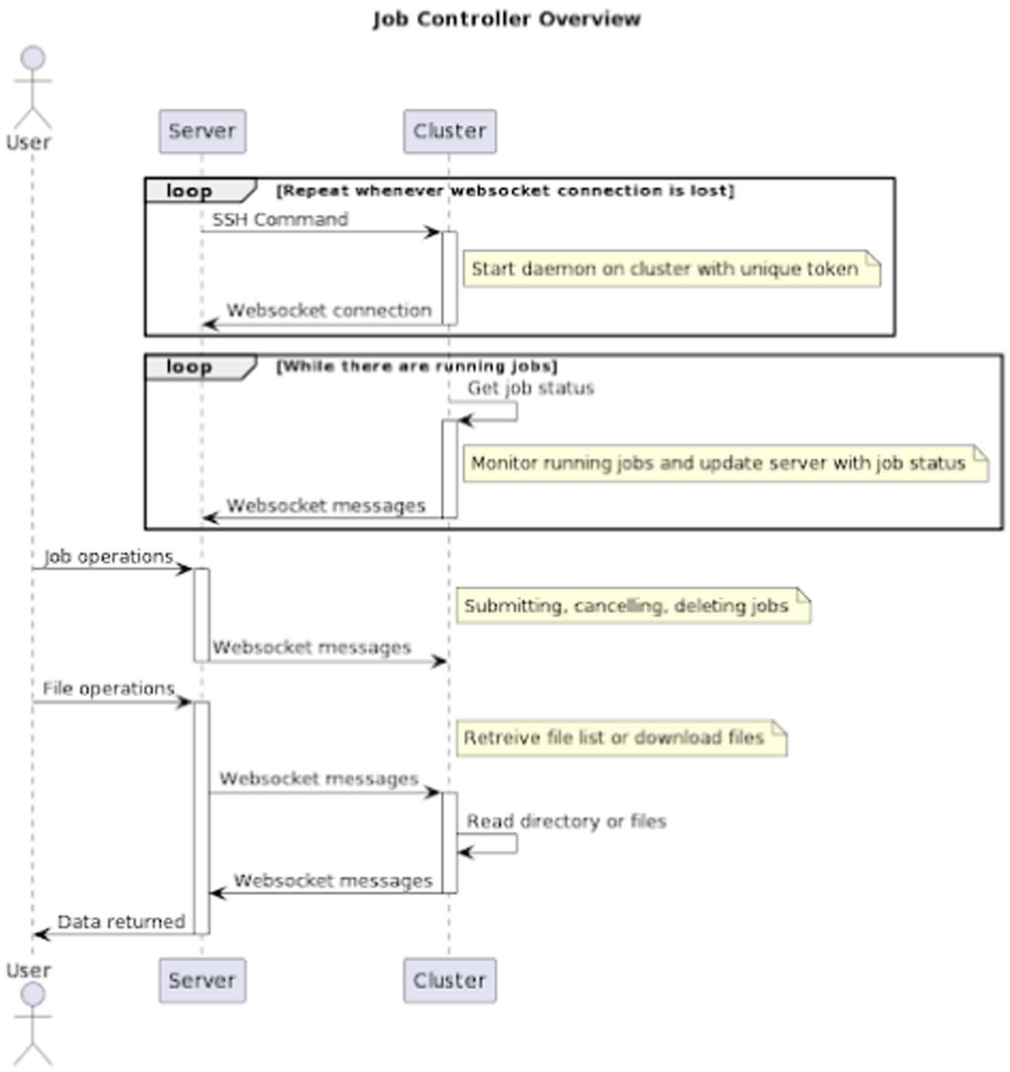

Job Controller Overview
User
Server
Cluster
loop [Repeat whenever websocket connection is lost]
SSH Command
Start daemon on cluster with unique token
Websocket connection
loop [While there are running jobs]
Get job status
Monitor running jobs and update server with job status
Websocket messages
Job operations
Submitting, cancelling, deleting jobs
Websocket messages
File operations
Retreive file list or download files
Websocket messages
Read directory or files
Websocket messages
Data returned
User
Server
Cluster

Figure 9. High-level infrastructure architecture diagram: `Kubernetes` cluster component diagram. The component diagram represents a high-level segregation of components hosted by the `Kubernetes` cluster. The virtual hosts represent the virtual machines that are included as part of the `Kubernetes` cluster. The control plane represents the atomic components allowing `Kubernetes` to be operational. Management tools are the workloads deployed for managing operations-related activities. This involves fundamental requirements such as security, networking, and storage-related administration. Application workloads include all custom applications developed for GWDC. At this time of writing, it includes applications directly involved with GWCLOUD and GWLAB, as another example.

contemporary practices for HPC facility management. A new solution was required.

The Job Controller is itself three discrete components: the Job Controller Server,[43] Job Controller Client,[44] and Bundles. The Job Controller Server is deployed in the `Kubernetes` infrastructure and exposes an API that can be used by various modules, including GWCLOUD, for submitting jobs, retrieving the status of jobs, canceling jobs, and retrieving file lists and downloading job files from remote clusters. The server is written in C++ for multithreaded performance and uses a MYSQL database for persisting information about jobs, their states, and for caching job file lists for complete jobs.

The Job Controller Client is written in PYTHON and runs as a daemon on all leveraged remote clusters, but can be deployed on any system supporting secure shell (SSH) communication with PYTHON 3 installed. It communicates directly with the server via a `WebSocket`[45] established when the client is initiated, which is instigated by the server via SSH. The client then forks itself to become a daemon and the SSH connection is dropped. This architecture has the advantage that the only communication needed by the remote cluster is a brief initiating inbound SSH interaction and HTTPS for any subsequent communication with the server. The server can direct the client to submit a new job, cancel a running job, or delete data relating to past jobs. The client tracks the state of running jobs via Bundles (described below) and reports job state updates to the server. Importantly, the client also provides the ability for the server to request a file list for a job in real time, and for the server to ask the client to send a job result file over the `WebSocket` for transfer to a user via browser or API. A single Job Controller Server may have many clients on many remote clusters.

Communication between the client and the server happens over one single `WebSocket` connection and is scheduled using a Multi Level Priority Queue:[46] an algorithm taken from operating system design. This allows higher-priority data (such as job file lists) to be sent first over the `WebSocket`, while

[43] https://github.com/gravitationalwavedc/gwcloud_job_server
[44] https://github.com/gravitationalwavedc/gwcloud_job_client
[45] https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API
[46] https://www.geeksforgeeks.org/multilevel-queue-mlq-cpu-scheduling/

lower-priority data (such as file transfers) are sent last or as "best efforts." This design keeps the client/server communication responsive for real-time events (such as when the user requests a file list for a job) at only a slight throughput cost to file transfers, for example. All communication is between the client and server.

If a `WebSocket` connection is dropped or broken, the client is terminated, and the server attempts to restart the remote client via SSH. If the SSH connection fails, the server will intermittently retry the SSH connection until it succeeds. This provides minimal downtime, and resiliency against cluster maintenance or transient connectivity issues between the client and server.

Bundles are the final component of the Job Controller. They contain the business logic required to prepare a job for submission, submit it, and to check its execution status. A single Job Controller Client may have many bundles, which could represent different projects (GWCLOUD, GWLAB, etc.) or different versions of runtime codes (e.g., BILBY) leveraged by them. A versioned history of bundles is maintained by the client to support the robust reproducibility of past jobs, if required. In the case of GWCLOUD, its bundle is responsible for rewriting the `ini` file for the Local Cluster hosting a job, downloading and storing supporting files, and tracking the state of the job for SLURM and CONDOR batch schedulers.

ORCID iDs

Paul D. Lasky https://orcid.org/0000-0003-3763-1386
Eric Thrane https://orcid.org/0000-0002-4418-3895

References

Abbott, B. P., Abbott, R., Abbott, T. D., et al. 2016a, PhRvL, 116, 241103
Abbott, B. P., Abbott, R., Abbott, T. D., et al. 2016b, PhRvL, 116, 061102
Abbott, B. P., Abbott, R., Abbott, T. D., et al. 2018, PhRvL, 121, 161101
Abbott, B. P., Abbott, R., Abbott, T. D., et al. 2019, ApJL, 882, L24
Abbott, R., Abbott, T. D., Abraham, S., et al. 2020a, ApJL, 896, L44
Abbott, R., Abbott, T. D., Abraham, S., et al. 2021b, ApJL, 913, L7
Abbott, R., Abbott, T. D., Abraham, S., et al. 2021c, SoftX, 13, 100658
Abbott, R., Abbott, T. D., Abraham, S., et al. 2021d, PhRvX, 11, 021053
Abbott, R., Abbott, T. D., Acernese, F., et al. 2020b, PhRvL, 125, 101102
Abbott, R., Abbott, T. D., Acernese, F., et al. 2021a, arXiv:2111.03606
Abbott, R., Abbott, T. D., Acernese, F., et al. 2023, PhRvX, 13, 011048
Ashton, G., Thrane, E., & Smith, R. J. E. 2019a, PhRvD, 100, 123018
Ashton, G., Hübner, M., Lasky, P., et al. 2019b, ApJS, 241, 27
Baibhav, V., Berti, E., Gerosa, D., et al. 2019, PhRvD, 100, 064060
Baiotti, L. 2019, PrPNP, 109, 103714
Biwer, C. M., Capano, C. D., De, S., et al. 2019, PASP, 131, 024503
Brown, T. 2008, Harvard Bus. Rev., 86, 84
Callister, T. A., Haster, C.-J., Ng, K. K. Y., Vitale, S., & Farr, W. M. 2021, ApJL, 922, L5
Cao, Z., & Han, W.-B. 2017, PhRvD, 96, 044028
Chatziioannou, K. 2020, GReGr, 52, 109
Chatziioannou, K., Cornish, N., Wijngaarden, M., & Littenberg, T. B. 2020, PhRvD, 103, 044013
Chia, H. S., Olsen, S., Roulet, J., et al. 2022, PhRvD, 106, 024009
Evans, M., Adhikari, R. X., Afle, C., et al. 2021, arXiv:2109.09882
Fuller, J., & Ma, L. 2019, ApJL, 881, L1
Galaudage, S., Talbot, C., Nagar, T., et al. 2021, ApJL, 921, L15
Gayathri, V., Healy, J., Lange, J., et al. 2022, NatAs, 6, 344
Gerosa, D., Pratten, G., & Vecchio, A. 2020, PhRvD, 102, 103020
Holz, D. E., & Hughes, S. A. 2005, ApJ, 629, 15
Hoy, C., Fairhurst, S., Hannam, M., & Tiwari, V. 2022, ApJ, 928, 75
Hoy, C., & Raymond, V. 2021, SoftX, 15, 100765
Kapadia, S., Caudill, S. J., Creighton, J. D. E., et al. 2020, CQGra, 37, 045007
Lackey, B. D., & Wade, L. 2015, PhRvD, 91, 043002
Landry, P., & Essick, R. 2019, PhRvD, 99, 084049
Lange, J., O'Shaughnessy, R., & Rizzo, M. 2018, arXiv:1805.10457
Lenon, A. K., Nitz, A. H., & Brown, D. A. 2020, MNRAS, 497, 1966
Littenberg, T. B., & Cornish, N. J. 2015, PhRvD, 91, 084034
Liu, X., Cao, Z., & Shao, L. 2020, PhRvD, 101, 044049
Lower, M. E., Thrane, E., Lasky, P. D., & Smith, R. J. E. 2018, PhRvD, 98, 083028
Maggiore, M., Van Den Broeck, C., Bartolo, N., et al. 2020, JCAP, 3, 050
Miller, S., Callister, T. A., & Farr, W. M. 2020, ApJ, 895, 128
Nielsen, J. 2000, Designing Web Usability (Dover, DE: Nielsen Norman Group)
Nitz, A. H., Kumar, S., Wang, Y.-F., et al. 2023, ApJ, 946, 59
Norman, D. 2013, The Design of Everyday Things: Revised and Expanded Edition (New York: Basic)
Olsen, S., Venumadhav, T., Mushkin, J., et al. 2022, PhRvD, 106, 043009
Payne, E., Talbot, C., & Thrane, E. 2019, PhRvD, 100, 123017
Pordes, R., , OSG Consortium, Petravick, D., et al. 2007, JPhCS, 78, 012057
Pratten, G., & Vecchio, A. 2021, PhRvD, 104, 124039
Pratten, G., García-Quirós, C., Colleoni, M., et al. 2021, PhRvD, 103, 104056
Reitze, D., Adhikari, R. X., Ballmer, S., et al. 2019, BAAS, 51, 035
Romero-Shaw, I. M., Lasky, P. D., & Thrane, E. 2019, MNRAS, 490, 5210
Romero-Shaw, I. M., Lasky, P. D., & Thrane, E. 2021, ApJL, 921, L31
Romero-Shaw, I. M., Lasky, P. D., Thrane, E., & Calderón Bustillo, J. 2020a, ApJL, 903, L5
Romero-Shaw, I. M., Talbot, C., Biscoveanu, S., et al. 2020b, MNRAS, 499, 3295
Roulet, J., Venumadhav, T., Zackay, B., Dai, L., & Zaldarriaga, M. 2020, PhRvD, 102, 123022
Schutz, B. F. 1986, Natur, 323, 310
Smith, R., Ashton, G., Vajpeyi, A., & Talbot, C. 2020, MNRAS, 498, 4492
Smith, R. J. E., & Thrane, E. 2018, PhRvX, 8, 021019
Sun, L., Goetz, E., Kissel, J. S., et al. 2020, CQGra, 37, 225008
Talbot, C., & Thrane, E. 2022, ApJ, 927, 76
Thrane, E., & Talbot, C. 2019, PASA, 36, E010
Vajpeyi, A., Smith, R., & Thrane, E. 2023, ApJ, 947, 10
Varma, V., Field, S. E., Scheel, M. A., et al. 2019, PhRvR, 1, 033015
Veitch, J., & Vecchio, A. 2010, PhRvD, 81, 062003
Veitch, J., Raymond, V., Farr, B., et al. 2015, PhRvD, 91, 042003
Vitale, S., Gerosa, D., Farr, W. M., & Taylor, S. R. 2022, Handbook of Gravitational Wave Astronomy (Berlin: Springer), 45
Vivanco, F. H., Smith, R., Thrane, E., & Lasky, P. D. 2020, MNRAS, 499, 5972
Vivanco, F. H., Smith, R. J. E., Thrane, E., et al. 2019, PhRvD, 100, 103009
Williams, D., Veitch, J., Chiofalo, M. L., et al. 2023, JOSS, 8, 4170
Wysocki, D., O'Shaughnessy, R., Wade, L., & Lange, J. 2020, arXiv:2001.01747