BAMS In Box

Cb-WoFS: Migrating the Warn-on-Forecast System to the Cloud

Joshua J. Martin,[a,b] Adam J. Clark,[b] Nusrat Yussouf,[a,b,c] Louis Wicker,[b] Pamela Heinselman,[b] Kent Knopfmeier,[a,b] Brian C. Matilla,[a,b] Patrick C. Burke,[b] and Salar Adili[d]

KEYWORDS: Numerical weather prediction/forecasting; Mesoscale models; Optimization

ABSTRACT: This article provides a brief, technical narrative of the WoFS journey to a cloud-based high-performance computing (HPC) system, including some of the technological challenges encountered and solutions found. Also, discussed are a few new components that are in development for cloud-based Warn-on-Forecast System (cb-WoFS), such as the cloud infrastructure project for managing resources, as well as a new web application that manages cb-WoFS runs. An important initial step in our cloud journey is containerizing all of the compiled applications, such as WRF, GSI, and EnKF and their dependencies like netCDF and MPI. With these applications compiled within an Apptainer container, WoFS can run on any local or cloud-based HPC cluster that supports MPI. Furthermore, an additional software layer was developed that creates and manages cloud vendor resources. This layer, which is referred to as the WoFS framework, contains the workflow required to run cb-WoFS, as well as management for other aspects of cb-WoFS (including but not limited to creation of HPC pools in the end-to-end workflow, runtime notifications, and database management). This additional layer was developed to separate the WoFS business logic from vendor-specific API calls. The WoFS framework exposes features through its service library, which is then referenced by the newly developed cb-WoFS web application and other cloud applications. This makes WoFS a complete end-to-end cloud-based application, where an administrator can launch a model run, manage resources, and view output all within a single web app.

DOI: 10.1175/BAMS-D-23-0296.1

Corresponding author: Joshua J. Martin, joshua.martin@noaa.gov

Manuscript received 20 November 2023, in final form 8 August 2024, accepted 29 August 2024

AFFILIATIONS: [a] Cooperative Institute for Severe and High-Impact Weather Research and Operations, University of Oklahoma, Norman, Oklahoma; [b] NOAA/OAR/National Severe Storms Laboratory, Norman, Oklahoma; [c] School of Meteorology, University of Oklahoma, Norman, Oklahoma; [d] Microsoft Corporation, Redmond, Washington

1. Introduction

The NOAA National Severe Storms Laboratory (NSSL) started the Warn-on-Forecast research program in 2009 with the goal of dramatically increasing lead times for hazardous convective weather within the watch to warning time frame (i.e., 0–6 h). By using a rapidly updating, storm-scale numerical weather prediction (NWP) ensemble modeling system to provide probabilistic predictions, forecasters could potentially shift from a warn-on-detection paradigm, which bases warnings on observations, to a warn-on-forecast paradigm that combines observations and NWP guidance. Through a robust research program at NSSL and its partner institutions over the past 15 years, this vision is coming to fruition. NSSL has developed a mature prototype known as the Warn-on-Forecast System (WoFS). WoFS is an on-demand, ensemble data assimilation (DA) and prediction system with 3-km grid spacing, which provides probabilistic storm-based guidance at 0–6-h lead times with output available at 5-min intervals. The current WoFS prototype is derived from version 3.9 of the Weather Research and Forecasting (WRF) Model, with a 900 km × 900 km movable domain. The DA analysis ensemble has 36 members which are updated every 15 min by assimilating surface, radar, and satellite observations using the ensemble Kalman filter (EnKF) algorithm within NOAA's Community Gridpoint Statistical Interpolation analysis system (GSI) software package. A diagram of this workflow can be seen in Fig. 1.

The basic concept of operations for WoFS is that when an area of potential hazardous weather risk is identified, a WoFS domain is centered over that region. Initial and boundary conditions are obtained from the current operational High-Resolution Rapid Refresh (HRRR; Dowell et al. 2022) system. Ideally, the ensemble state is first "spun-up" for 2 h preceding convective initiation. After ensemble spinup, forecasts are initialized from the first 18 ensemble members of the DA system every 30 min; forecasts at the top (bottom) of the hour integrate out to 6 h (3 h). The WoFS typically runs for 12–15 h per event. Forecasts are available on a modern web viewer 25–30 min after the analysis time. For more information on WoFS, including configuration details, postprocessing, visualization, and verification approaches, as well as user-focused research, see Heinselman et al. (2024).

Since WoFS is designed to focus on high-impact weather prediction associated with convective hazards, there may be periods when the system is not needed. Operational NWP centers are often designed to allocate computational resources for all operational prediction systems to run on a daily basis. Running WoFS on traditional high-performance computing (HPC) would require setting aside portions of the HPC system that could remain idle for long periods of time, wasting valuable resources. Furthermore, because WoFS also generates a large suite of probabilistic output fields at high temporal resolution every half hour, a combination of HPC for the ensemble prediction system and a tightly integrated postprocessing and display system is required for rapid dissemination of the output. As a result, WoFS has been developed as an on-demand system hosted on a cloud-based platform that can provide both the HPC and web services needed for timely use. The associated

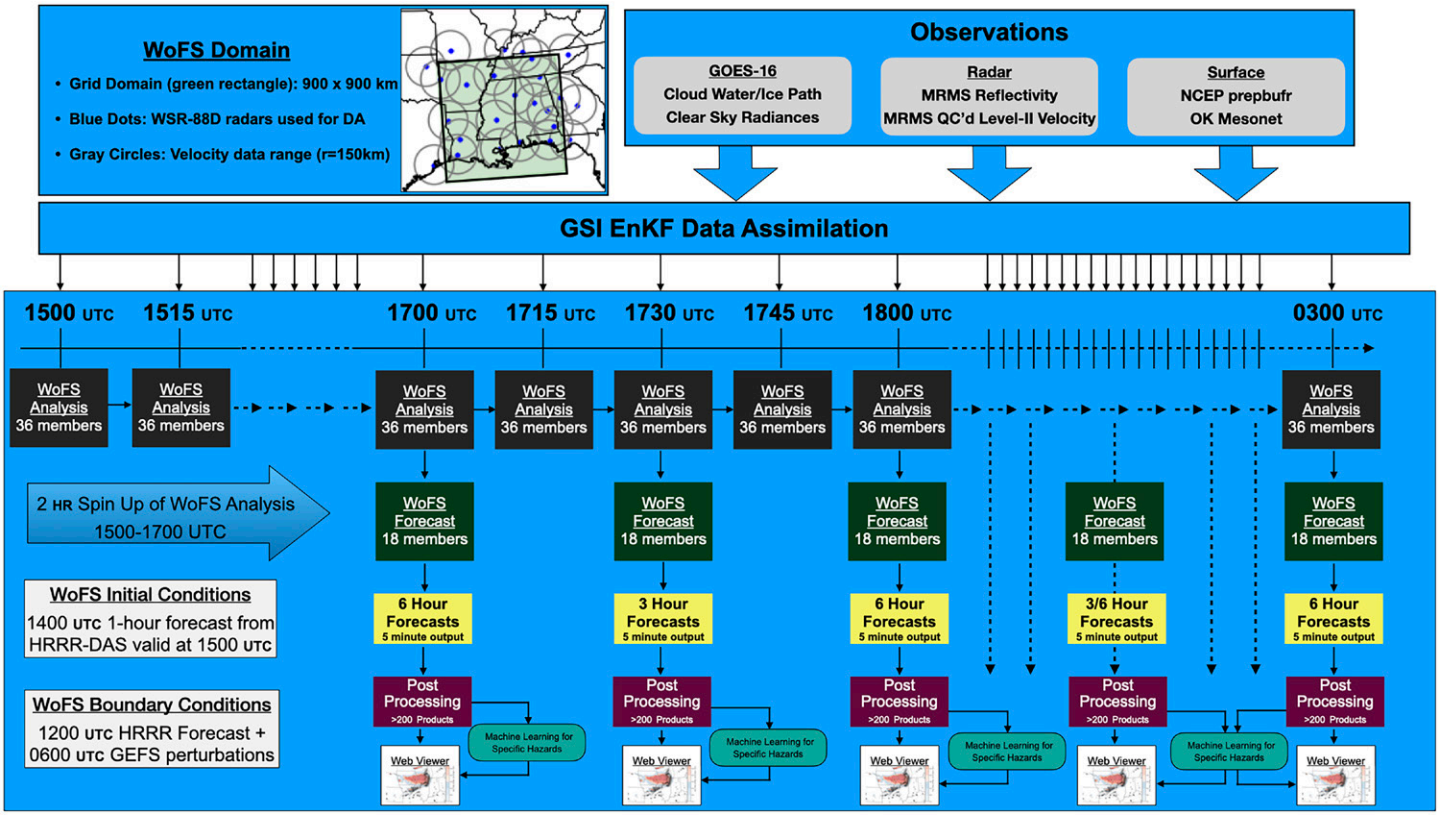

FIG. 1. WoFS workflow diagram (Heinselman et al. 2024). WoFS analysis begins at *T* + 5 min, with forecasts beginning at *T* + 15 min and postprocessing finishing by *T* + 30 min (*T* + 15 min) for 6-h (3-h) forecasts.

pay-as-you-go pricing model offers significant benefits to a system like WoFS, with reduced overall cost and access to the latest hardware advancements (see Table A1 in the appendix for definitions to common cloud computing terms).

2. NWP in the cloud

A recently published report estimated significant cost savings with running the Icosahedral Nonhydrostatic (ICON) weather model in the cloud rather than on-premise (Prill and Eser 2022). The report estimated that running ICON twice a day for a full year would cost less than $50,000. This was a maximum rough estimate that does not take into account the decreasing cost of services seen in the cloud computing market as a result of continued growth. In contrast, the average hardware cost to run national or regional models for 1 year is over $1 million, with the additional cost of operations and management exceeding $200,000. The estimates do not include the costs associated with downtime, hacker attacks, server maintenance, or the costs of system replacement which typically need to be updated every 5 years (Prill and Eser 2022). The cost savings associated with WoFS would be even greater because it is not needed on a daily basis. Other studies support the use of cloud computing to reduce HPC costs for NWP (Siuta et al. 2016; Powers et al. 2021; Celeski 2020; Freeman 2020). Chui et al. (2019) discussed how to further increase the cost efficiency on the cloud for NWP by using preemptible resources and file compression. In this article, we will discuss the system that NSSL built on Microsoft Azure that includes many of these technologies, including preemptive node usage, summary files with compression, and other storage cost savings.

3. WoFS in the cloud

Through 2021, WoFS was run on NSSL's in-house HPC system.[1] In early 2020, NSSL began exploring the cloud and evaluated the differences between Amazon Web Services (AWS) and Azure

[1] The original on-premise system was a Cray XC-30, with each node containing 24 cores. We utilized 36 nodes for cycling, 54 nodes for forecasts, and several 120-core Intel workstations for postprocessing.

by testing its NWP applications on the vendor platforms. After extensive testing, Microsoft Azure was selected as the cloud provider that best met NSSL's requirements for a cloud-based Warn-on-Forecast System (cb-WoFS). There were several factors that contributed to the vendor selection, including NWP performance, cloud resource cost, and development cost. At the time, the Azure platform was chosen because of its AMD hardware for HPC, which was both faster and cheaper than the equivalent Intel offerings. In addition, Azure's public cloud holds a FedRAMP High P-ATO[2] compliance accreditation. This accreditation allows for a secure and compliant deployment of cb-WoFS in an environment that is rich with the latest technology and large-scaled infrastructure (for more information on Azure and FedRAMP, see Microsoft 2023g). This facilitates a smoother transition from research to operations in the future.

[2] Refer to Table A1 in the appendix for a glossary of terms referenced in this paper.

4. WoFS HPC

The HPC aspect of cb-WoFS was built on Azure's scheduler-as-a-service offering, Azure Batch,[3] which automatically manages pools, tasks, and scaling (Microsoft 2024a). The Batch Service will automatically create a pool, or collection, of InfiniBand connected nodes, loaded with the desired OS that has been tuned for MPI workloads. With the Batch Service as the scheduler, an autoscale formula was developed that gives preference to low-cost preemptive[2] nodes, but when the number of preempted nodes reaches a certain threshold, automatically drop them and switch to the exact number of dedicated nodes required for the model run (for more information on Batch autoscaling, see Microsoft 2023e). This cost-effective solution happens behind the scenes and without any user interaction. Building on top of this platform reduced development time and system complexity because scheduler responsibilities are managed by the Azure service rather than the user. This design offers numerous benefits. By offloading more tasks to the cloud provider, the system's overall complexity is reduced. When the cloud provider takes charge of specific responsibilities, such as scheduler related activities, it leads to a reduction in the amount of code required in the application. Furthermore, aligning the system with platform services offered by a cloud provider enhances the user's ability to leverage the full spectrum of features provided by that service. Thus, platform-as-a-service (PaaS) offerings took priority over infrastructure-as-a-service (IaaS) offerings because of their feature richness, scalability, and overall reduction in system complexity.[2] This decision did, however, require more upfront development work, as most system components had to be updated to "fit" into the PaaS model.

[3] Refer to Table A2 in the appendix for descriptions of Azure resources referenced in this paper.

5. WoFS DevOps and architecture

The WoFS software stack consists of many distinct components, each residing in its dedicated GitHub repository. For example, the code that preprocesses satellite observations for DA is a stand-alone application with its own repository, as is the case with radar preprocessing, WRF, GSI, EnKF, web apps, and others. With every WoFS component already on GitHub, the first migration step was to create a Terraform infrastructure project that brings everything together by describing what cloud services are needed and which application runs on which service. Terraform is a powerful tool that allows the user to express infrastructure as human-readable code (HashiCorp 2024). With the entire cb-WoFS infrastructure expressed as human-readable code in one project, NSSL can have its infrastructure code managed and source controlled in GitHub right alongside WoFS application code. This gives NSSL the ability to quickly create everything that is required to run cb-WoFS on any Azure cloud subscription, simplifying the deployment process of WoFS. Terraform is cloud neutral and provides plugins for all the major cloud service providers. The user must specify the cloud provider in the terraform project and which service offerings to configure—and cloud vendor service offerings are not all the same.

For example, although AWS Lambdas and Azure Functions[3] are similar in nature, they have subtle cost and feature differences that may be important to a user's project.

The Terraform architecture project is also used to define the continuous integration/continuous deployment strategy. There are many tools available for building a DevOps[2] environment, and we opted for Azure DevOps due to its seamless integration with Microsoft Entra ID[3] (formerly Azure Active Directory), GitHub, built-in private PyPI[2] artifact feeds, and project planning features (Microsoft 2022). With the Terraform DevOps plugin, we were able to define a build pipeline for every application required to run WoFS. Each build definition in the architecture project points to a GitHub repository and a YAML[2] file for instructions on how to build the repository. The YAML file instructs Azure DevOps to download the source code, build the Docker image, and publish the image to NSSL's internal Azure Container Registry.[3] For more information on defining a build pipeline in YAML, see Microsoft (2023f). All WoFS Docker applications are automatically built and deployed to NSSL's private Azure Container Registry using a simple build definition YAML file. With the entire system containerized and on GitHub, NSSL has the ability to rerun a WoFS case using the configuration from any point in time, making the system reproducible and easy to maintain. In addition, many of the platform services responsible for executing these Docker applications are configured to automatically pull the latest build from the private registry when updated.

6. WoFS core applications

With an architecture project in place, as well as the build definitions for each project, the various types of WoFS software packages, or applications, are now described. As we have already discussed, many of the applications were converted into Docker applications that could be invoked with simple command line arguments. Some of the Python projects, however, were developed into packages that could be installed in an environment using Python's pip[2] module. A Python package is ideal for situations where the code can be reused in other projects. A DevOps build pipeline, similar to the ones defined above, pulls the project from GitHub, builds the environment, and upon successful completion of the project's unit tests, creates a Python installable package. The package is then deployed to a private PyPI feed so that other internal build processes can install it into a container, or install it directly into a researcher's custom environment. An example of this would be the postprocessing application that analyzes ensemble member output and generates graphics. This application would need to run in some fashion on the cloud but is also useful to researchers for simple and consistent plotting utilities. A Python package is a perfect fit for this type of application, and automatically deploying to an Azure DevOps[3] private PyPI feed can easily be configured (Microsoft 2023a).

Python packages and Docker containers are not complicated, but what about NWP applications? At the time of development, there were clear performance advantages to building message passing interface (MPI) applications within Apptainer (formerly Singularity; see Apptainer 2021) as opposed to Docker. For WoFS, the NWP model and compiled utilities (such as GSI, EnKF, and others) are packaged into a single Apptainer image. These utilities often have software dependencies, such as netCDF, that require the same compiler. Therefore, packaging the compiled applications into a single image keeps things simple. The build pipeline for these applications is defined similarly to the method outlined above, but it uses the Apptainer command-line interface (CLI) to build the container from a build recipe in source control. There are other advantages to using Apptainer instead of Docker, other than just performance. For one, there is no need to publish your image to a Docker registry. Instead, the entire image is contained within a single file that can be passed around and executed on any Apptainer-enabled system. After the build pipeline generates the WoFS NWP image, the new image is versioned and uploaded to Azure Storage[3] as an Azure Batch application, ready for immediate use (for more information on Batch Applications, see Microsoft 2023b).

7. Putting it all together

A database-driven web application was developed to unify all aspects of the application so that configuring, launching, and viewing a model run could all be accomplished from within a single user interface. This required a new, generic software layer to manage the cloud workflow, with a separate implementation layer to interface with the Azure API. For the local HPC system, the functionality existed in the form of shell scripts that would move files to various component directories and execute workflows. For the cloud, this workflow logic was developed into a development framework that would call into the necessary cloud APIs to accomplish these same tasks within cloud provider services. This development framework is used by the web application to trigger a model run, as seen in Fig. 2.

The development framework creates an Azure Batch Task for every unit of work seen in Fig. 1. For example, when launching a new forecast for a cb-WoFS model run, the development framework creates an Azure Batch Task for every member at every initialization. It supplies everything required for the execution of that task to Azure Batch, including the run command, and URLs for the initial conditions and boundary conditions. Downloading prerequisite files is handled entirely by the Azure Batch service (Microsoft 2021). The run command is a standard MPI command to launch the weather model contained within the Apptainer image, like the following, "mpirun -n X -ppn Y apptainer exec –bind $PWD /path/to/image /path/to/wrf." Fortran namelists are handled a little differently than standard storage methods. Instead of saving a namelist to Azure Storage for every cycle, member, and application, an internal web application was developed where components could make HTTP requests for dynamically generated namelists. The Batch Task creates the namelist on the fly by making an HTTP request to the internal web application with the cb-WoFS run ID, forecast member, and initialization time. Once the task has been submitted to the Batch Service, the Azure Batch Scheduler handles the rest: reserving available nodes from the pool, downloading the necessary files from storage, running the MPI tasks, and uploading any output back to storage. For the 3-km WoFS, the "Hot"[2] storage tier has proven sufficient, but a higher-resolution WoFS would require a more performant file system, such as Azure Managed Lustre (Microsoft 2023c).

8. Postprocessing

Postprocessing the model forecast output in a timely manner is essential for cb-WoFS. Hundreds of products are created from the postprocessing package, an example of which can be seen in Fig. 3.

To take full advantage of the cloud and its scalability, we developed postprocessing around an Azure Storage Queue for scalability (see Microsoft 2023d for more information on Azure

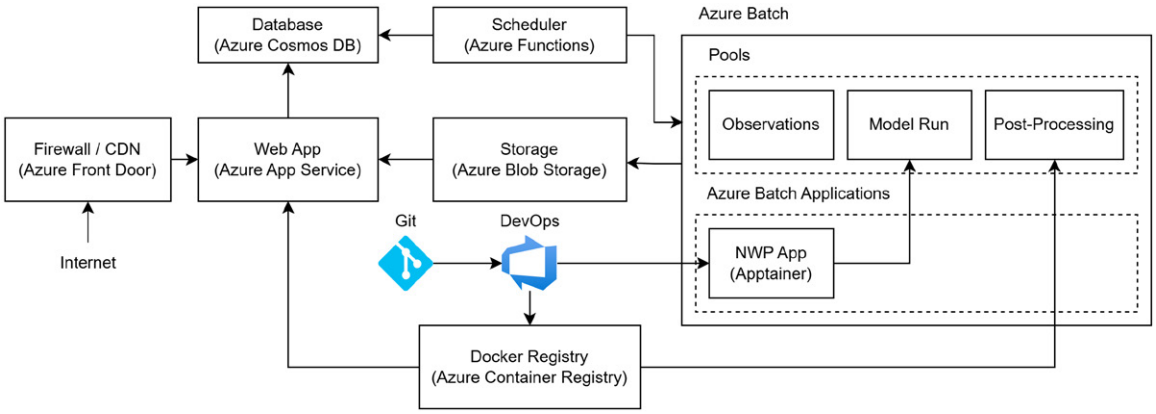

Fig. 2. Basic architectural diagram of cb-WoFS and its CI/CD workflow. As applications are updated, DevOps tests, builds, and deploys the containerized component into an internal Docker registry or as a Batch application. When a model run is launched from the website, the scheduler generates real-time tasks within Azure Batch.

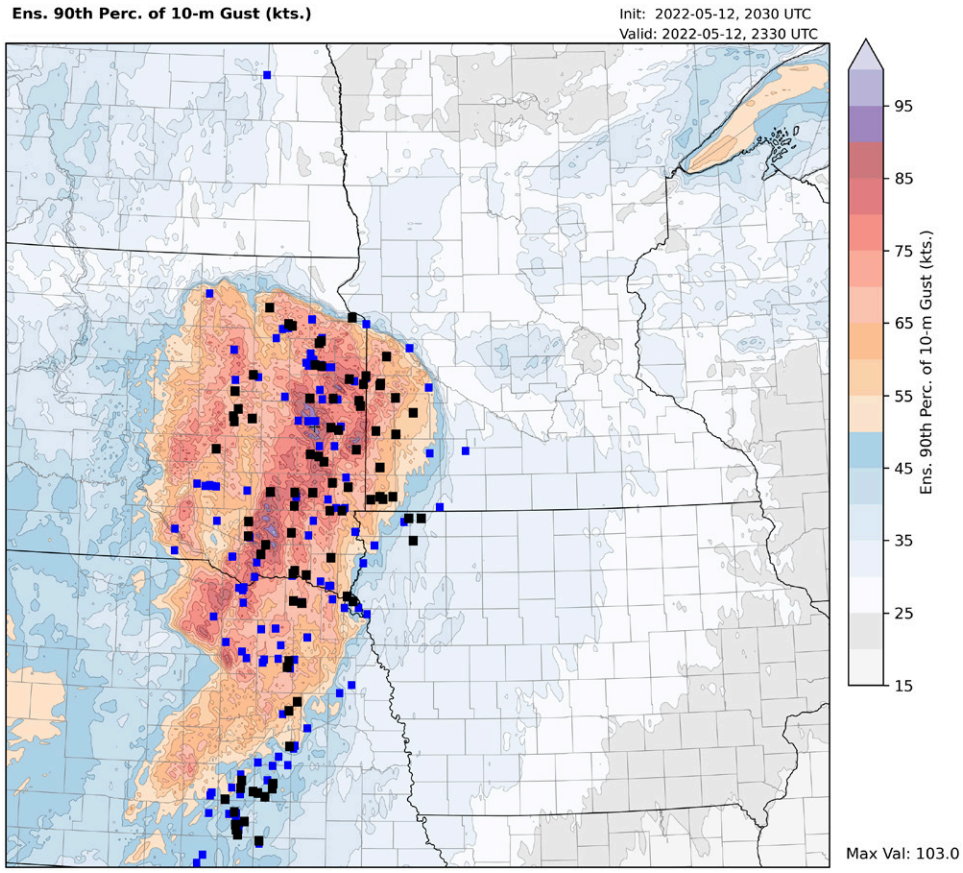

Fig. 3. Cb-WoFS 90th percentile of maximum surface wind on 5 Dec 2022, overlaid with blue (black) markers indicating reported severe wind gusts of 50 kt (1 kt ≈ 0.51 m s^{-1}) (65 kt) or greater (Burke et al. 2022). This highlights the accuracy of cb-WoFS in predicting the general region, magnitude, and timing of the severe weather event.

Storage Queues). As forecast output is generated, a new message is enqueued that contains all necessary information for postprocessing, such as the cb-WoFS run ID and URLs of the forecast output. Meanwhile, a scalable array of simultaneously running postprocessing tasks monitors the queue and waits for work. The Docker application, equipped with the WoFS postprocessing Python application, retrieves messages from the queue and creates a "summary file," which is a netCDF file containing specific model variables from all members for a single time step. These files use lossy compression on most variables to reduce file size. Web graphics are then generated from the corresponding summary file and uploaded to Azure Storage.

With postprocessing dockerized, the application can scale according to the number of domains that are running. Initially, we ran these tasks on Azure App Service,[3] which offers built-in scalability and Python packages that enable the user to easily specify a function that is automatically invoked whenever a message is queued in an Azure queue (for more information, see Microsoft 2024b). While effective, this approach incurred costs that exceeded our desired budget. In the end, it was more cost effective to create a wrapper application that monitors the queue manually and then create long-running Docker tasks within Azure Batch. This method also makes the application more portable; it can run anywhere. It could just as easily run in Kubernetes or on a local machine, as it is no longer built specifically for Azure App Service. As postprocessing generates graphics, the output is immediately uploaded to storage, and a database is updated with the model run's progress. Users following the model run on cb-WoFS are updated with the latest output via web sockets, a technology standard that enables data to be pushed from a web server to connected browser clients.

9. Results

With WoFS now on the cloud, the WoFS application is almost entirely a self-contained cloud-native application. Owing to its design, a new cloud environment can be created and destroyed in only a matter of minutes. In addition, individual WoFS components are modular and scalable, which enables NSSL to run multiple WoFS domains over the United States simultaneously. A single web app facilitates the initiation, monitoring, and viewing of model runs, simplifying the process through an intuitive user interface (Fig. 4).

This interface grants cb-WoFS administrators the capability to initiate a WoFS run without requiring formal training in HPC. With WoFS on the cloud, NSSL does not have to pay for any unused resources and also has access to the latest HPC technology as cloud vendors upgrade their systems[4] to the latest hardware. For example, in 2022, Azure upgraded their AMD EPYC Milan processors to Milan-X at no additional cost to the user, boosting cb-WoFS performance by as much as 20% without any changes to the system. When a WoFS run is launched, the HPC environment is built and begins processing in less than 10 min. Upon completion, the environment is removed and the only remaining associated cost with the model run is data storage. The average cost for a 12-h cb-WoFS run, not including storage costs, is less than $1000. Cloud-based WoFS was successfully demonstrated during NOAA's Hazardous Weather Testbed (HWT) Spring Forecasting Experiment in 2022 and 2023 (Clark et al. 2023a,b) and will continue to be used in the foreseeable future. Cloud computing

[4] One interesting aspect of cloud computing for an application that requires the same amount of computational resources is that over time (~2–3 years) the cost of running becomes cheaper as the application moves from running on the most expensive to less expensive systems.

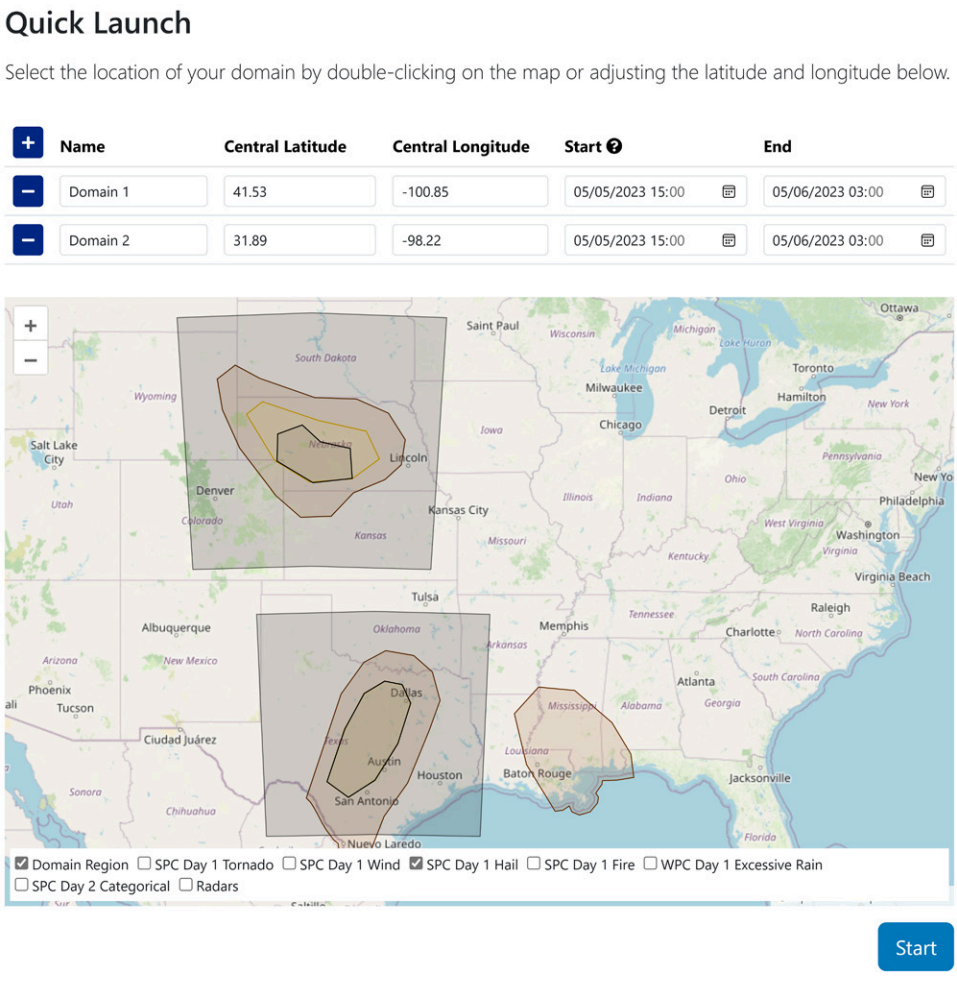

Fig. 4. Screenshot of cb-WoFS Quick Launch screen, demonstrating how the screen would have appeared on 5 May 2023. Each box highlights the region of each domain, overlayed with the Storm Prediction Center's Day 1 1200 UTC large hail outlook.

provides an unmatched advantage for sporadic yet scalable applications, offering on-demand resources that expand to meet surges in usage, ensuring cost efficiency and optimal performance without the burden of maintaining continuous infrastructure. Finally, the development of cloud-based WoFS is a unique example of a successful public–private partnership that is enhancing NOAA's mission, while at the same time helping the private sector understand user needs to better direct their product roadmap.

Acknowledgments. This work is supported by the NOAA/Office of Oceanic and Atmospheric Research under NOAA-University of Oklahoma Cooperative Agreement NA21OAR4320204, U.S. Department of Commerce. The authors would like to thank Monte Flora and the three anonymous reviewers, whose thoughtful feedback greatly improved the quality of this article.

APPENDIX
Cloud Computing Terminology

Table A1 provides a brief definition of many commonly used cloud computing terms referenced in this article. Table A2 describes the Azure services used by cb-WoFS.

Table A1. A glossary of common terms.

Term	Definition
AMD	Advanced micro devices
Apptainer (formerly singularity)	Containerization platform primarily used in HPC environments. For additional information on containers, see Docker.
CI/CD	Continuous integration/continuous deployment strategies are key practices within DevOps. These practices are designed to streamline and automate the software delivery process.
DevOps	DevOps, a combination of "development" and "operations," is a set of practices and principles with the goal of improving collaboration between development and operations. The practices usually include a CI/CD strategy.
Docker	An open-source platform for creating, running, and deploying applications within a container. Containers enable developers to package an application with all of its dependencies and runtime environment. An image is a snapshot of an application and its dependencies. A container is a running instance of an image.
GitHub	GitHub.com is a web-based platform for version control and collaborative software development, built around Git. It hosts repositories, facilitates project management, and provides tools for code collaboration and version tracking.
Hot storage tier	Azure's storage tiers provide a range of performance and pricing choices, from high-performance "hot" storage to cost-effective "cold" storage for less frequently accessed data.
IaaS	Infrastructure-as-a-Service is a cloud computing model that offers virtualized computing resources like servers, storage, and networking, allowing users to build and manage their own IT infrastructure.
Kubernetes	Kubernetes is an open-source container orchestration platform that automates the deployment, scaling, and management of containerized applications.
PaaS	Platform-as-a-Service is a cloud computing model that provides a platform for customers to develop, run, and manage applications without dealing with the complexity of infrastructure management.
P-ATO	Provisional Authority to Operate is an authorization pathway for a cloud service provider to operate with the federal government. The authorization indicates that the cloud service provider has met the required security standards developed by the Federal Risk and Authorization Management Program (FedRAMP).
Preemptible resource	A cloud resource (virtual machine or otherwise) that can be provisioned at a significantly reduced cost with the caveat that the cloud provider can terminate it without notice.
Pip	Python's pip module is a package management system that allows you to install and manage software packages written in Python.
PyPI	Python Package Index, a repository of Python software packages.
YAML	YAML is a human-readable data serialization format that uses a simple, text-based structure to organize and store data. It is commonly used for configuration files due to its readability and ease of use.

Table A2. Description of Azure services referenced in this manuscript.

Azure service	Description
App service	Platform-as-a-Service for building, deploying, and scaling web apps and APIs, offering features like auto-scaling, continuous integration, and a variety of programming languages.
Batch	Designed for running large-scale parallel and HPC applications efficiently by automatically scheduling and managing the execution of tasks across a pool of virtual machines. It simplifies the process of scaling compute resources to meet demand, enabling users to run complex workloads without the need to manually configure or manage infrastructure.
Container registry	Managed Docker container registry service that allows users to store, manage, and deploy container images and artifacts in a secure and scalable manner.
DevOps	Provides development tools for planning, developing, testing, and delivering software, integrating CI/CD pipelines, version control, and collaborative workspaces.
Entra ID (formerly Active Directory)	Identity and access management service that provides secure access to resources and applications through single sign-on, multifactor authentication, and conditional access.
Functions	Serverless compute service that enables users to run event-driven code without managing infrastructure.
Storage	Provides scalable, durable, and secure storage options for data objects, files, disks, and queues, accessible via REST APIs and client libraries.

References

Apptainer, 2021: Community announcement. Accessed 14 February 2024, https://apptainer.org/news/community-announcement-20211130/.

Burke, P. C., and Coauthors, 2022: Collaborating to increase warning lead time using the Warn-on-Forecast System. *30th Conf. on Severe Local Storms*, Santa Fe, NM, Amer. Meteor. Soc., 44, https://ams.confex.com/ams/30SLS/meetingapp.cgi/Paper/407184.

Celeski, S., 2020: Forecasting the weather with AWS cloud. Presentation, AWS Public Sector Summit Online, 19 pp., https://pages.awscloud.com/rs/112-TZM-766/images/PartnerSolutions_Forecasting%20the%20weather%20with%20AWS%20Cloud_Maxar.pdf.

Chui, T. C. Y., D. Siuta, G. West, H. Modzelewski, R. Schigas, and R. Stull, 2019: On producing reliable and affordable numerical weather forecasts on public cloud-computing infrastructure. *J. Atmos. Oceanic Technol.*, **36**, 491–509, https://doi.org/10.1175/JTECH-D-18-0142.1.

Clark, A. J., and Coauthors, 2023a: The third real-time, virtual spring forecasting experiment to advance severe weather prediction capabilities. *Bull. Amer. Meteor. Soc.*, **104**, E456–E458, https://doi.org/10.1175/BAMS-D-22-0213.1.

——, and Coauthors, 2023b: The first hybrid NOAA hazardous weather testbed spring forecasting experiment for advancing severe weather prediction. *Bull. Amer. Meteor. Soc.*, **104**, E2305–E2307, https://doi.org/10.1175/BAMS-D-23-0275.1.

Dowell, D. C., and Coauthors, 2022: The High-Resolution Rapid Refresh (HRRR): An hourly updating convection-allowing forecast model. Part I: Motivation and system description. *Wea. Forecasting*, **37**, 1371–1395, https://doi.org/10.1175/WAF-D-21-0151.1.

Freeman, J., 2020: Cloud high performance computing. *Presentation, Annual R&D Workshop*, Online, Australia Bureau of Meteorology, 18 pp., http://www.bom.gov.au/research/workshop/2020/Talks/Justin-Freeman.pdf.

HashiCorp, 2024: What is terraform? Accessed 14 February 2024, https://developer.hashicorp.com/terraform/intro.

Heinselman, P. L., and Coauthors, 2024: Warn-on-Forecast System: From vision to reality. *Wea. Forecasting*, **39**, 75–95, https://doi.org/10.1175/WAF-D-23-0147.1.

Microsoft, 2021: Use multi-instance tasks to run Message Passing Interface (MPI) applications in Batch. Accessed 14 February 2024, https://learn.microsoft.com/en-us/azure/batch/batch-mpi.

——, 2022: Overview of services. Accessed 14 February 2024, https://learn.microsoft.com/en-us/azure/devops/user-guide/services?view=azure-devops.

——, 2023a: Quickstart: Publish and consume Python packages with Azure Artifacts using the command line (CLI). Accessed 14 February 2024, https://learn.microsoft.com/en-us/azure/devops/artifacts/quickstarts/python-packages?view=azure-devops.

——, 2023b: Deploy application packages to compute nodes with Batch application packages. Accessed 14 February 2024, https://learn.microsoft.com/en-us/azure/batch/batch-application-packages.

——, 2023c: What is Azure Managed Lustre? Accessed 14 February 2024, https://learn.microsoft.com/en-us/azure/azure-managed-lustre/amlfs-overview.

——, 2023d: What is Azure Queue Storage? Accessed 14 February 2024, https://learn.microsoft.com/en-us/azure/storage/queues/storage-queues-introduction.

——, 2023e: Create a formula to automatically scale compute nodes in a Batch pool. Accessed 14 February 2024, https://learn.microsoft.com/en-us/azure/batch/batch-automatic-scaling.

——, 2023f: YAML vs Classic pipelines. Accessed 14 February 2024, https://learn.microsoft.com/en-us/azure/devops/pipelines/get-started/pipelines-get-started.

——, 2023g: Federal Risk and Authorization Management Program (FedRAMP). Accessed 21 March 2024, https://learn.microsoft.com/en-us/azure/compliance/offerings/offering-fedramp#azure-and-fedramp.

——, 2024a: What is Azure Batch?. Accessed 14 February 2024, https://learn.microsoft.com/en-us/azure/batch/batch-technical-overview.

——, 2024b: Azure Queue storage trigger for Azure Functions. Accessed 14 February 2024, https://learn.microsoft.com/en-us/azure/azure-functions/functions-bindings-storage-queue-trigger.

Powers, J. G., K. K. Werner, D. O. Gill, Y.-L. Lin, and R. S. Schumacher, 2021: Cloud computing efforts for the Weather Research and Forecasting Model. *Bull. Amer. Meteor. Soc.*, **102**, E1261–E1274, https://doi.org/10.1175/BAMS-D-20-0219.1.

Prill, F., and C. Eser, 2022: ICONIC – ICON in the Cloud. Reports on ICON, issue 009, DWD and Max-Planck-Institute for Meteorology, 30 pp., https://www.dwd.de/EN/ourservices/reports_on_icon/pdf_einzelbaende/2022_09.pdf;jsessionid=966FC7B096F82A67DB3477967F9BEC0A.live11042?__blob=publicationFile&v=13.

Siuta, D., G. West, H. Modzelewski, R. Schigas, and R. Stull, 2016: Viability of cloud computing for real-time numerical weather prediction. *Wea. Forecasting*, **31**, 1985–1996, https://doi.org/10.1175/WAF-D-16-0075.1.