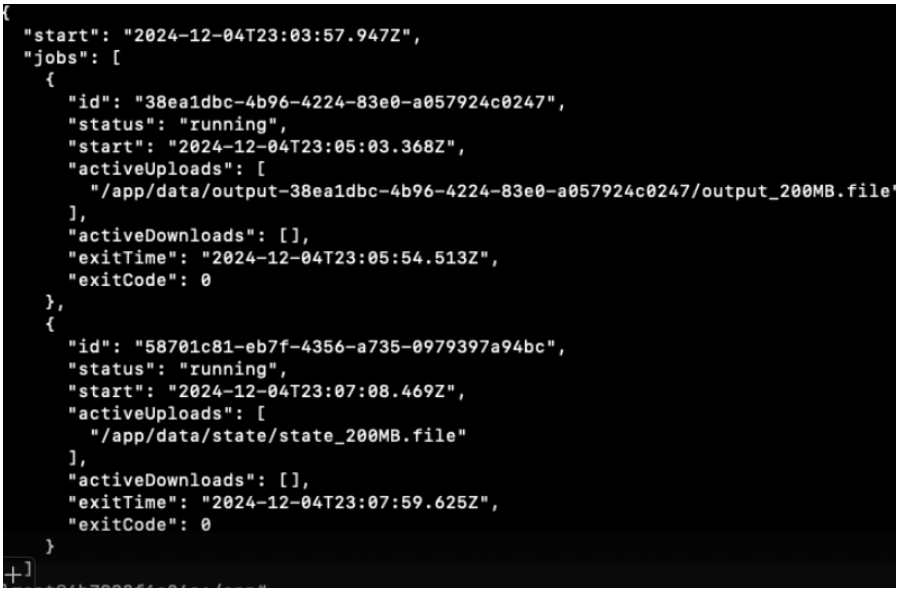
	Key Specs - Kelpie 0.5.0	
1	The Kelpie worker checks for new jobs every 10 seconds when the node is idle.	
2	The Kelpie platform marks a job as COMPLETED when the application exits with a status of zero and the Kelpie worker successfully uploads the job data (output and state).	
3	When the application exits with a non-zero status on a node, the Kelpie worker will be ‘reporting job failed’ and the job will be retried by other nodes and this node will be banned from this job by the Kelpie platform. After 3 failed attempts (the infrastructure failures are not included in the count), the Kelpie platform considers a job FAILED.	An application may temporarily fail on a node due to issues such as insufficient RAM or VRAM, or incompatible CUDA versions.
4	A job can be run indefinitely without returning results explicitly, with support for unlimited node reallocations, including the application calling the IMDS Reallocate API and nodes going down unexpectedly.	
5	Once a job is initially retrieved and processed by a node, the job status transitions from PENDING to RUNNING. Regardless of any interruptions or non-zero exits that may occur afterward, the status remains RUNNING until the job is ultimately marked as COMPLETED or FAILED. As a result, a RUNNING job may be waiting in the queue after a node failure or if the app exits with a non-zero status. When a Kelpie worker retrieves a job, additional information, including heartbeat signals, is used to select appropriate jobs from the queue.	
6	After saving the final data to the output folder, the application exits with a status of zero. Meanwhile, the Kelpie worker continues uploading the data in the background. If the upload fails, the Kelpie worker will be ‘reporting job failed’ (equivalent to the application exiting with a non-zero status). The incomplete job, which remains in the RUNNING state, then becomes available for other nodes to process, provided the number of attempts is less than or equal to 3.	
7	After being banned from 5 jobs, a node will be reallocated by the Kelpie worker (invoking the IMDS Reallocate API).	
8	The Kelpie platform retains jobs regardless of their status—PENDING, COMPLETE, or FAILED—though you can delete a job using the Kelpie API.	
9	To enable the auto scaling, the current solution is to add SaladCloud to the organization. The Kelpie platform will then use the granted access and permissions to adjust the replica count of a container group as needed.	

10	Ensure that the Kelpie's built-in data management feature meets your requirements: -It downloads all data specified in the job from the cloud to the instance when the job starts. -It monitors the designated local folders during job execution and uploads any changes to the cloud. -It uploads all files from the specified local folder to the cloud once the job is completed. To support flexible data management (such as selectively downloading part of data when a job is run), consider building an AWS SQS-based solution and implementing your own custom data management. You can also use Salad Kelpie exclusively as a job queue, bypassing its data sync functionality and implementing your own data management. This can be achieved by including sync: {} in the job definition. Creating a container group with a single instance to run each simulation is also a viable approach for larger tasks that require many hours or even days to complete. You can utilize environment variables to specify the locations (e.g., cloud storage) for the job's input and output for each group instance, eliminating the need for a job queue.	
11	The Kelpie worker enables chunked, asynchronous, and concurrent downloads and uploads. During job execution, any modified and created files in the designated folders will be detected, divided into 10MB chunks, and uploaded by the Kelpie worker, with up to 8 chunks processed simultaneously. If changes exceed the uploading throughput, both new files and old files with distinct filenames will be uploaded simultaneously; however, any new changes to files that are already being uploaded will be ignored. When the application exits with a status of zero, the Kelpie worker copies the job output to a temporary folder for uploading. This enables the application to start processing a new job while the outputs from previous jobs are uploaded in the background. The timeout for upload and download have been removed. To improve application efficiency and reliability, you can further filter Salad nodes to ensure they have sufficient download and upload bandwidth, as well as sufficient throughput to your data store.	
12	The Kelpie Job ID is passed to the application via the environment variable - KELPIE_JOB_ID, which can be used in the application logs, along with the job creation jobs for troubleshooting.	
13	The Kelpie worker maintains a state file that tracks in-progress downloads and uploads, allowing the application to check the status if needed. The location of	

	this file is provided to the job via the environment variable KELPIE_STATE_FILE.  	In this example, the first job has already been completed, and the Kelpie worker is in the process of uploading its output. Meanwhile, the second job is still in progress, with the Kelpie worker actively uploading its running state. If a backlog of uploads occurs, the application can respond by adjusting its runtime parameters (e.g., modifying the saving interval) or reallocating the node to resolve the issue.