dis·co

Datasheet

Overview

Dis.co is a serverless computing platform that makes it easy to run your code on any number of machines. It’s a fast and cost-effective solution to long-running jobs.

You can use the web front-end interface to drag and drop a Python script to be launched, and data files to go with it. Bringing in 100 data files means the job can be parallelized on up to 100 machines. With the command line tool, you get the same ease of use in the terminal or bash script. You can do the same with the Python SDK, but in a more structured programming language. It’s also possible to use `discomp`, which is a drop-in replacement for the Python `multiprocessing` library, unlocking potentially massive acceleration with one small change to the code.

Dis.co offers a variety of deployments to fit your needs:

- Dis.co Cloud - A completely managed service where you get scalable compute without having to think about servers.
- Your Cloud - Connect instances from your own AWS, Google Cloud, Azure, and Packet infrastructure. With support for multiple cloud platforms, Dis.co optimizes the utilization of all your resources.
- On-Prem - The Dis.co agent and orchestrator lets you manage jobs on your machines and personal devices.

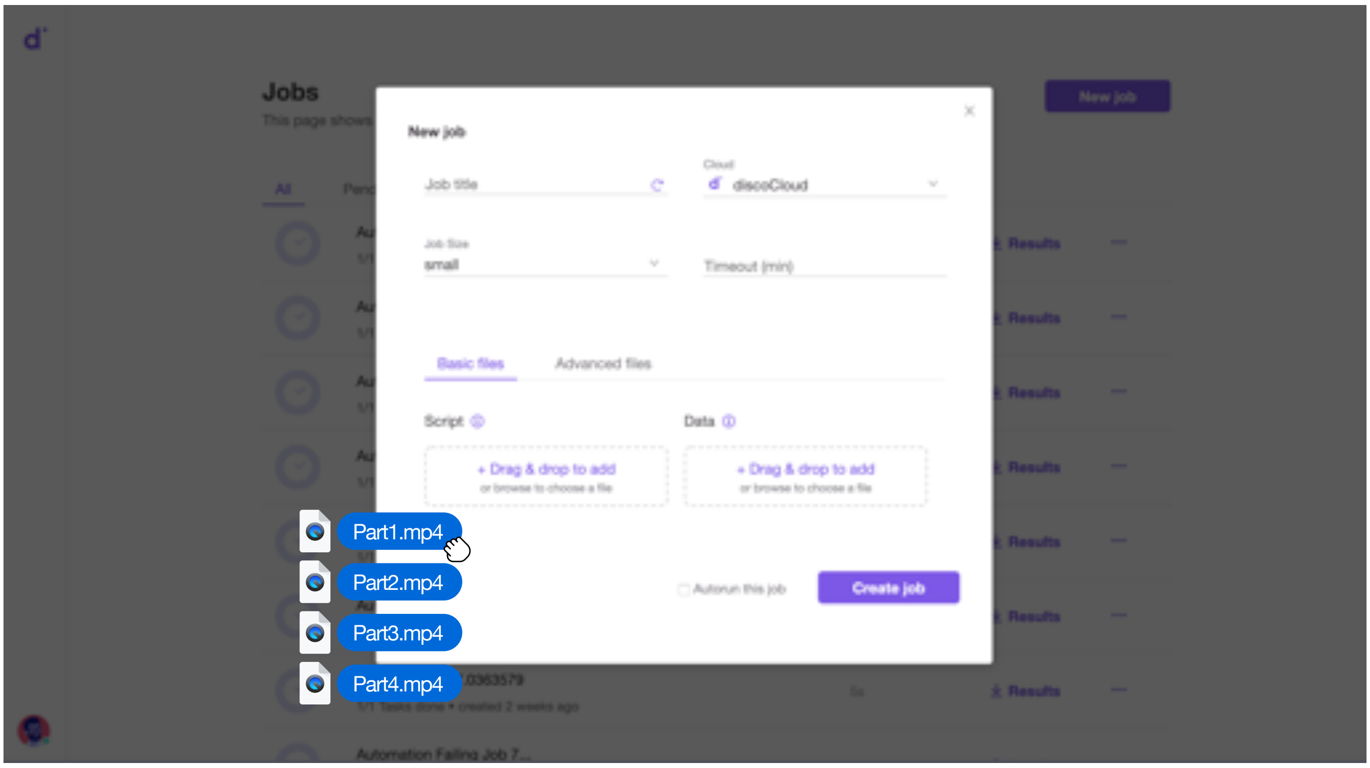
How It Works

Everything that runs on Dis.co is a job, and every job has its own script and data files. A task is spun up for every data file and is concretely run on one machine. After setting up a job, it can be run many times, without having to re-upload data files. The Dis.co Scheduler is used to route jobs to a network of Dis.co agents - which can be on a variety of devices, and infrastructure providers.

A number of serverless compute solutions are used today, including AWS Lambda, Google Cloud Platform Cloud Functions, Azure Functions. Each solution has its own command-line tools, API, Web UI, and gotchas. Each solution is configured dramatically differently and has its own limitations for execution time, memory, and runtime environment. These configurations and limitations are so complicated that every customer ends up having to build infrastructure to work around them. At Dis.co, after building many of these workarounds to existing batch computing systems, we wished for a simpler way and built it.

The following example will take all of the video part files in a folder, and send them to the cloud to be processed:
disco add --name process_video --script analyze.py --input "part*.mp4"

Notice that using * we could process any amount of video files that were in our current directory and named “part01.mp4”, “part02.mp4”, etc. To create the same job using the Web UI, drag the script and these mp4 files into your browser as pictured below. The experience using the Python SDK is similar. All dis.co APIs have been designed with developer ease in mind.



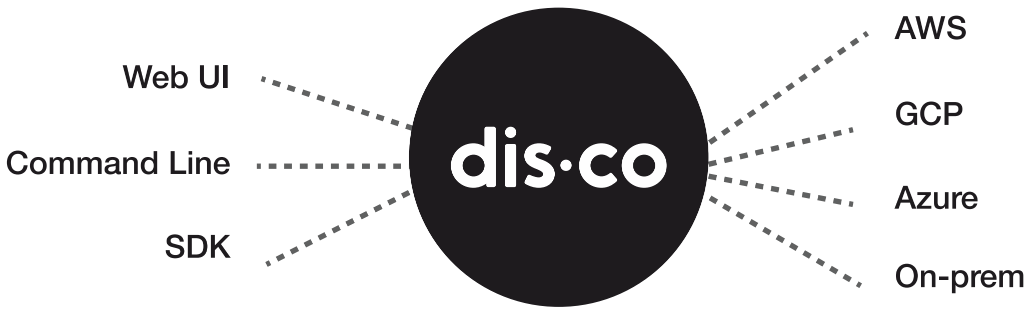
Cloud
discoCloud
Job Size
small

When the job completes, you can download the results as a zip file. The CLI, SDK, and Web UI provide a way to download all the results at once. Inside each task zip file, you’ll find standard output, standard error, and any other files that were written to `/local/run-result`. Using the SDK, the results can be returned as Python objects. Within less than a minute, we’ve started a job that will launch remote processes, auto-scale infrastructure, transfer code, assign data, and return results. That’s why Dis.co to is the simplest way to start a batch job.

Features

- **Parallel Execution, Auto-Scaled**
  If you're not already scaling out execution - your jobs are needlessly taking too long. Automated parallelization and scaling is far more efficient than manual resource allocation.
- **Serverless**
  No servers to provision, patch, manage, or monitor. Pay only for the compute time you use. You won't have to remember to turn the expensive machine on and off ever again.
- **Quick Integration**
  The UI, SDK, and APIs have been designed to be as simple as possible without sacrificing expressiveness.
- **No Learning Curve**
  Use existing codebase and CI/CD tools - Parallelize your Python projects without learning multi-processing. Works with Jenkins, CircleCI, TravisCI, and Pipelines.
- **Zero Dependency Management**
  Bring your own Docker images. Or let us build one based on your requirements.
- **Multiple Interfaces**
  Run jobs in web or command-line interface. Monitor your jobs from a dashboard with a modern design.
- **Modern Hybrid Cloud**
  Leverage existing infrastructure - on-prem and cloud - as one resource. Auto-scaling infrastructure with nothing to maintain. Avoid vendor-lock and single points of failure with a hybrid or multi-cloud configuration.
- **Smart Scheduler**
  Stop building home-grown scheduling and orchestration tools. Free up DevOps time.
- **No Arbitrary Limits**
  No max execution time, no max concurrent executions. You're paying for the compute time, so why impose artificial limitations on a serverless system?

Simplifying Hybrid and Multi-Cloud Operations



Tech Specs

Security

Customer code and data is copied to the agents for execution purposes only. Code and data is deleted from the agents as soon as the task terminates. All data transfers between Disco servers to agents are done over HTTPS (TLS). We collect task, machine and agent metadata to improve and enable intelligent resource management and scheduling capabilities. All the user information captured by Dis.co is outlined in the privacy policy.

If you'd like to integrate Dis.co with your own cloud provider, you'll need to run a Dis.co onboarding script, providing the Dis.co cloud account access to your own cloud. This script will:

- **Generate a designated bucket providing Dis.co full access to this bucket, and only to it.**
- **Generate a VPC in order to perform actions on instances on your cloud.**
- **Generate an IAM role providing Dis.co cloud account access to instances running on your cloud.**

Parallelization Levels

The reason we use t-shirt sizes at dis.co is that we found users often over and under provision resources. We're building a real-time predictor that would make better guess on what kind of resources a task needs, so it is best to use t-shirt sizes as a hint for the scheduler rather than a hard constraint.

T-shirt size has no influence on the parallelization factor of a job. A user can create a "small" job with 100 tasks and a "large" job with a single task. The level of parallelism is set by a combination of dis.co configuration and hosting cloud settings. For example, the cloud provider's scaling limitations can prevent dis.co from instantiating more machines. While these definitions will change in the future, the following table is what you should expect when specifying a t-shirt size.

T-shirt size	AWS	Azuret	GCP	Packet
S	c5.large	Standard_F2s_v2	n1-highcpu-2	t1.small
M	c5.xlarge	Standard_F4s_v2	n1-highcpu-4	t1.small
L	c5.2xlarge	Standard_F8s_v2	n1-highcpu-8	c2.medium.x86

Compatibility Matrix

Supported Clouds

Cloud Name	GUI Supported	CLI Support
AWS	✓	✓
GCP	✓	✓
Azure	✓	✓
Packet	✓	✓

Supported operating systems

Client OS	Agent	CLI and SDK Support
Ubuntu 18.04	✓	✓
Centos 7.7	✓	✓
Docker 19.03	✓	✓
Windows		✓
MacOS		✓

Get Started

Sign up now at https://dis.co/contact to see a demo of Dis.co and start your free trial. See for yourself how Dis.co delivers faster results, improves performance, and scales your processing power.