Provision-as-a-Service: Automating data center operations with Airflow at Cloudflare

CLOUDFLARE

Jet Mariscal
SRE, Edge @ Cloudflare

CLOUDFLARE'S MISSION:

Help build a better Internet

Cloudflare is an intelligent, integrated global cloud network that delivers security, performance, and reliability for all your Internet infrastructure, people and connected devices.

CLOUDFLARE

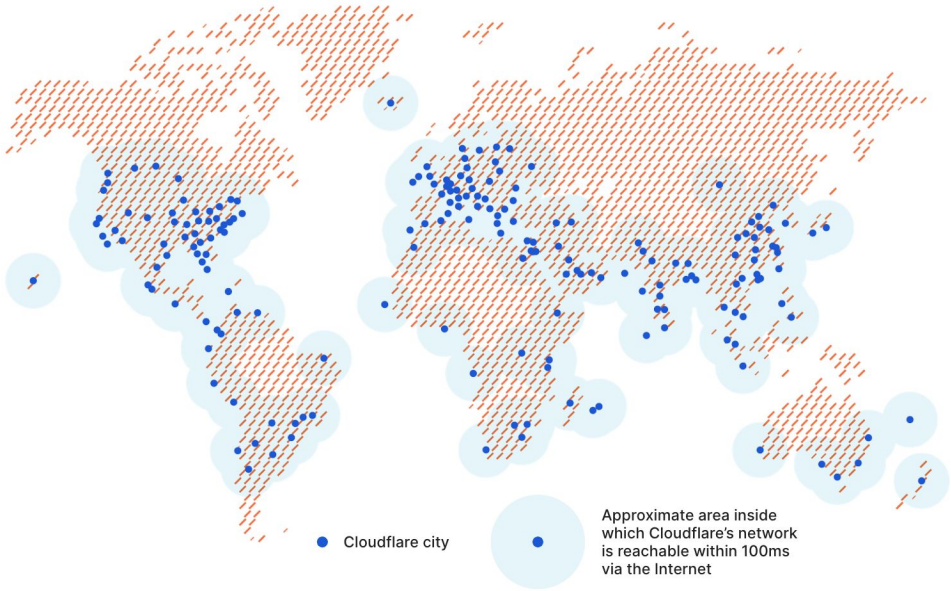
CLOUDFLARE'S MISSION:

Help build a better Internet

Cloudflare protects and accelerates any Internet application online without adding hardware, installing software, or changing a line of code.

CLOUDFLARE

Cloudflare's network operates at massive scale



~25M
Internet properties

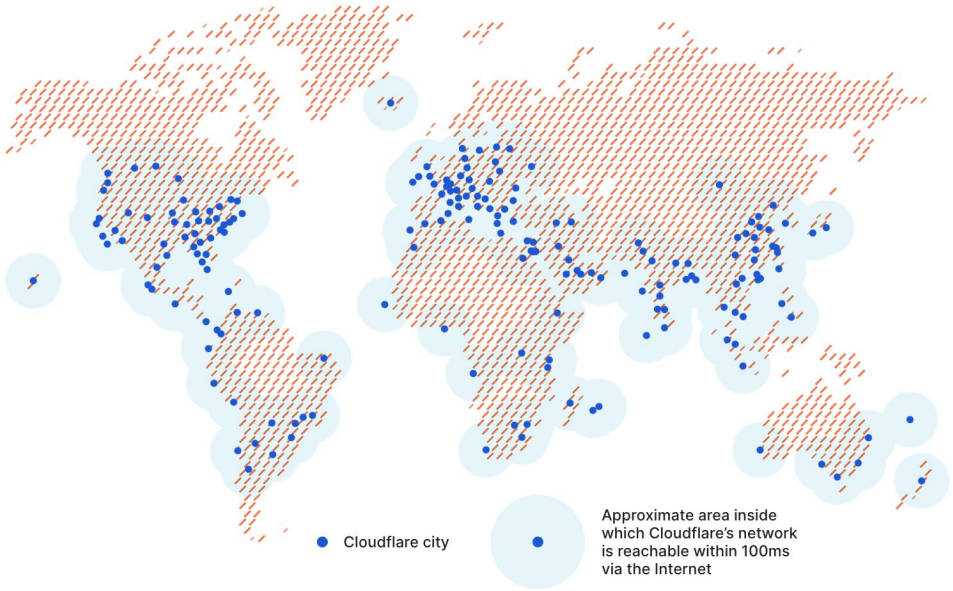
200+
Cities and 100+ countries

99%
Of the Internet-connected population in the developed world is located within 100 milliseconds of our network

17%
Of the Fortune 1000 are paying Cloudflare customers

CLOUDFLARE

Cloudflare's network operates at massive scale



25M

HTTP requests per second served on average, 30M+ at peak

9.4M

DNS queries per second, about 811 billion queries per day, and 24 trillion queries a month

70B

Cyber threats blocked each day in Q1'21

CLOUDFLARE

www.cloudflare.com

blog.cloudflare.com

cloudflare.tv

What is Provisioning?

CLOUDFLARE

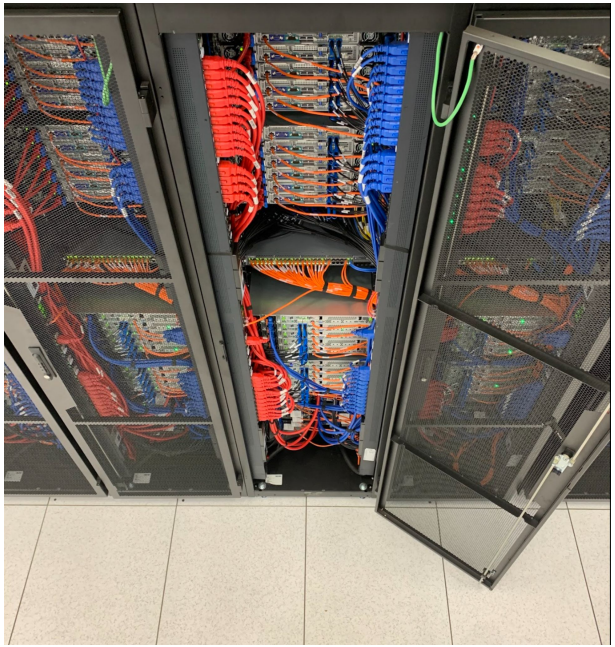
Provisioning: Expansions and Decommissions

Expansion is the process of adding new machines to expand the capacity of a data center.

Decommission is the process of permanently removing machines for retirement in a data center.

CLOUDFLARE

Provisioning is complex



Connecting new Cloudflare servers to our network used to be so complex, in large part because of the amount of manual effort required and careful coordination between Data Center and Infrastructure Operations, Network Operations, and SREs.

CLOUDFLARE

Manual Provisioning: a process that can only scale so far

Engineers used to carefully follow steps from an extremely detailed standard operating procedure (SOP) document, often copying command-line snippets and pasting it into terminal windows.

CLOUDFLARE

Manual Provisioning: slow, error-prone, and very inefficient

- logging in to remote hosts via SSH
- lots of copy/pasting commands to run
- launching web browsers to view Grafana and other internal dashboards

Manual Provisioning: tedious, time-consuming, and does not scale

Simultaneous expansions and/or decommissions
became very challenging.

CLOUDFLARE

Provision-as-a-Service: Automation with Apache Airflow

- Totally eliminated the need of using SSH
- Guaranteed consistency, compared to any manual actions
- Democratized the provisioning process
- Faster and safer expansions and decommissions
- Eliminated toil

Cut by 90% the amount of time our team spent on mundane operational tasks.

CLOUDFLARE

Replacing manual steps with an API-call equivalent

1. Login to a remote system.
2. Copy and paste the command in the terminal.
3. Replace the router placeholder in the command snippet with the actual value.
4. Execute the command.

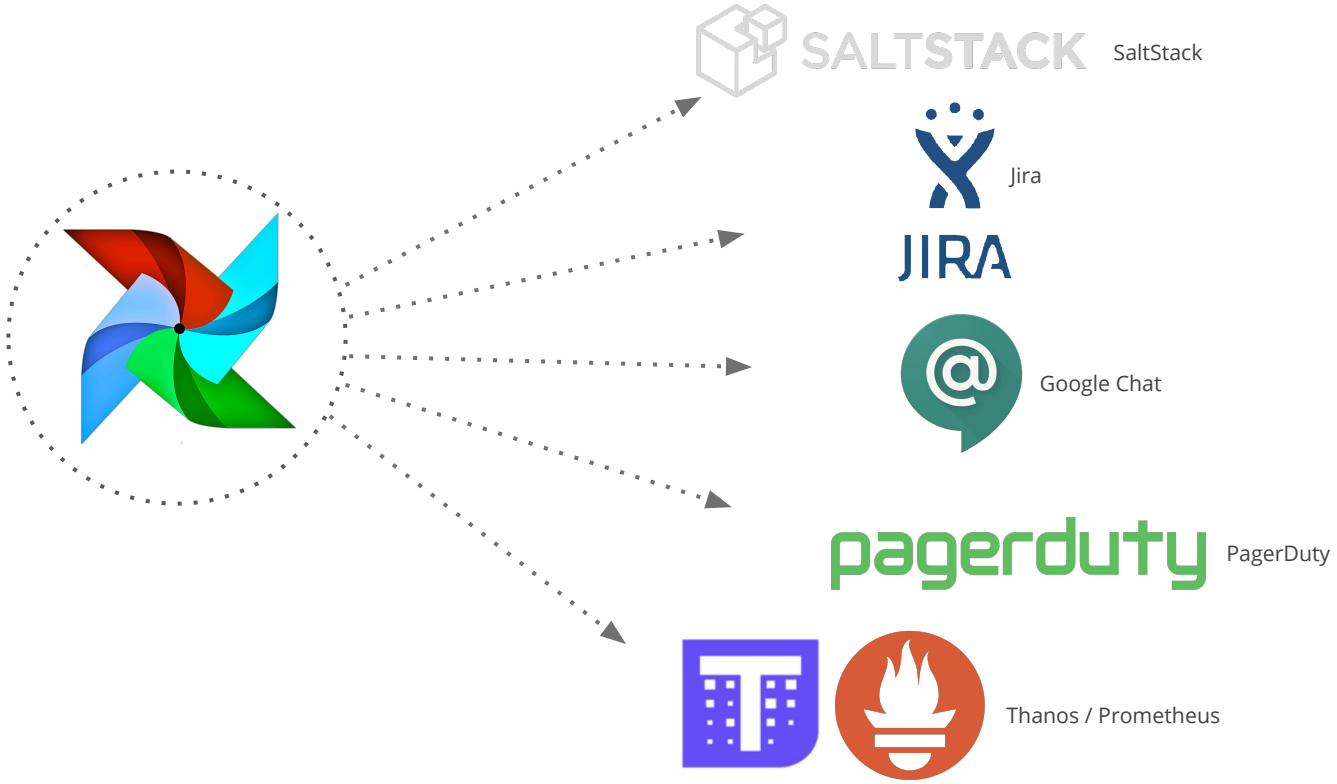
```
enable_anycast = builder.wrap_class(AsyncSaltAPIOperator)(
              task_id='enable_anycast',
              target='{{ params.netops }}',
              function='cmd.run',
              fun_kwargs={'cmd': 'salt {{ get_router(params.colo_name) }} '
                                 'anycast.enable --out=json --out-indent=-1'},
              salt_conn_id='salt_api',
              trigger_rule='one_success')
```

CLOUDFLARE

Manual steps transformed into a feature-packed automation

- Failure Handling
- Logging and Notifications
- Jinja templating
- Macros
- Task management via UI

Custom Operators to integrate with other systems



CLOUDFLARE

Adapting tasks for preconditions and human intervention

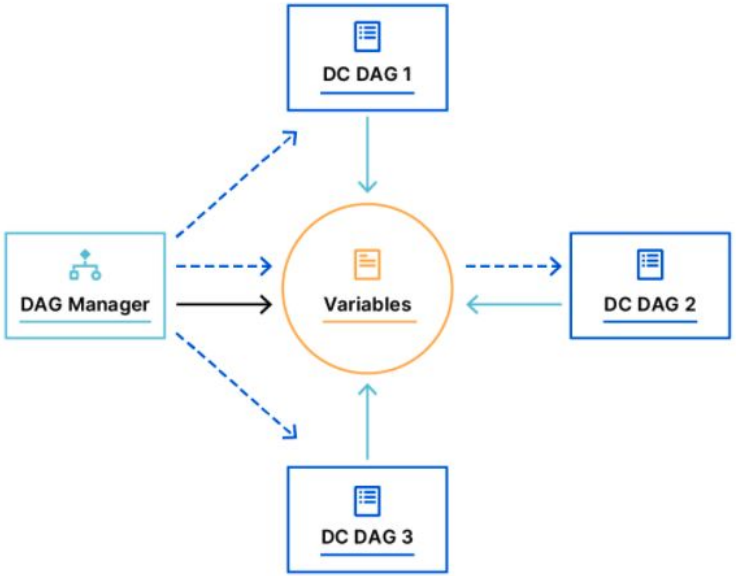
Using sensors to set dependencies between tasks or even DAGs, so that one does not run until the dependency has been met.

```
verify_node_dns = builder.wrap_class(DNSSensor)(
            task_id='verify_node_dns',
            zone=domain,
            nodes_from='{{ to_json(run_ctx.globals.import_nodes_via_mpl) }}',
            timeout=60 * 30,
            poke_interval=60 * 10,
        mode='reschedule')
```

```
verify_jira_input = builder.wrap_class(InputSensor)(
            task_id='verify_jira_input',
            var_key='jira',
            prompt='Please provide the Change Request ticket.',
            notify=True,
            require_human=True)
```

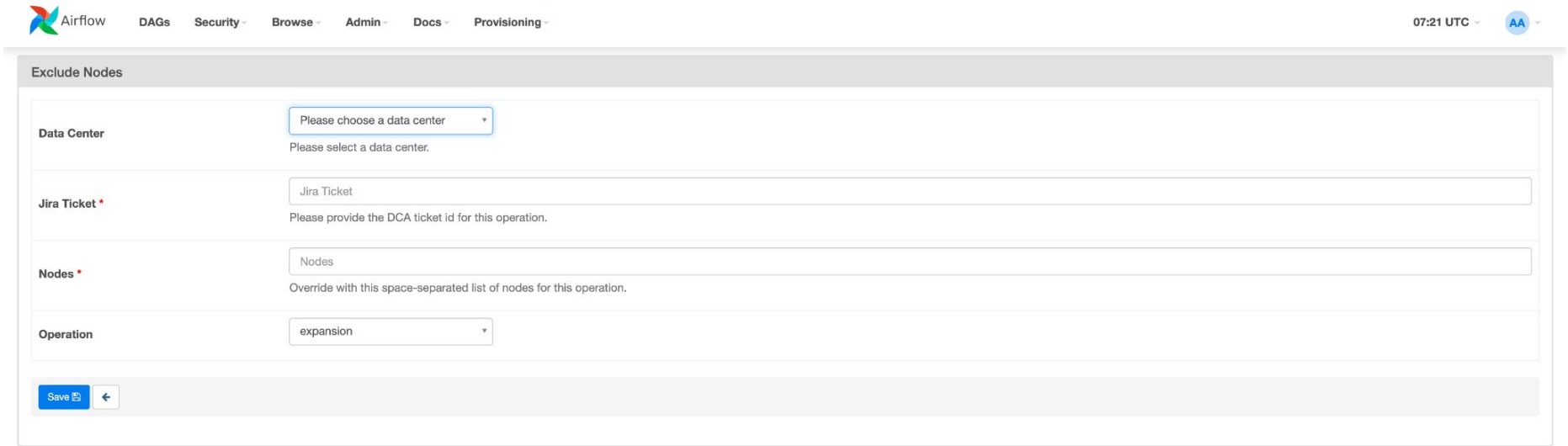
CLOUDFLARE

Accepting inputs and responding to human interventions



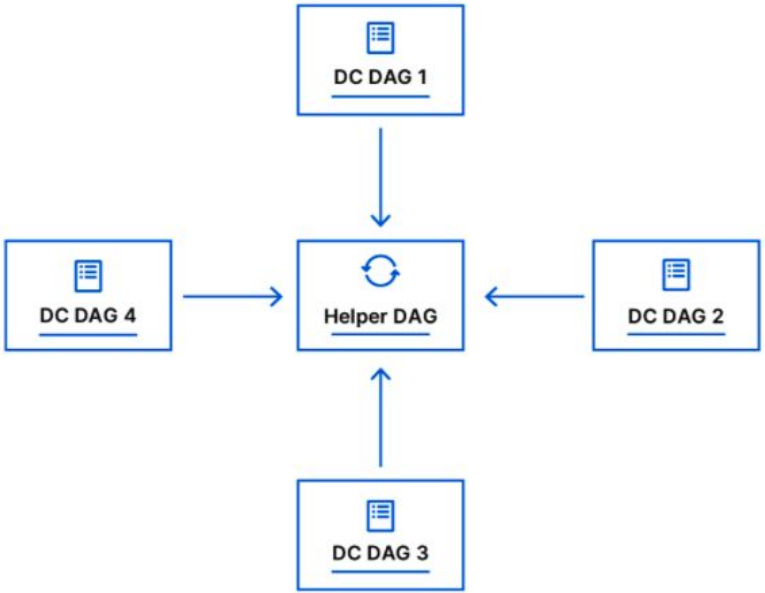
CLOUDFLARE

Custom forms for accepting user inputs



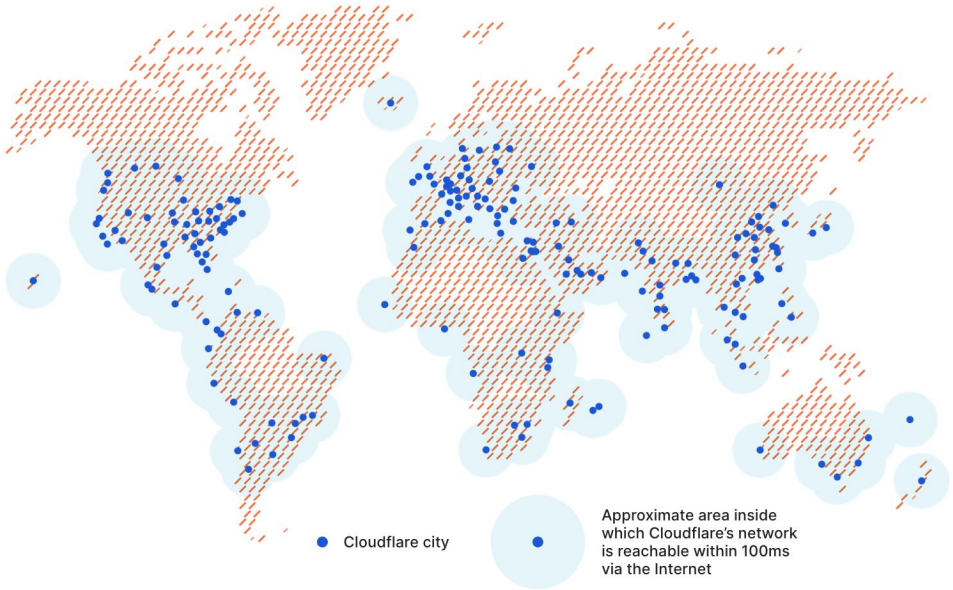
CLOUDFLARE

Solving complex workflows with Branching and Multi-DAGs



CLOUDFLARE

Creating DAGs that scale and executing tasks at scale

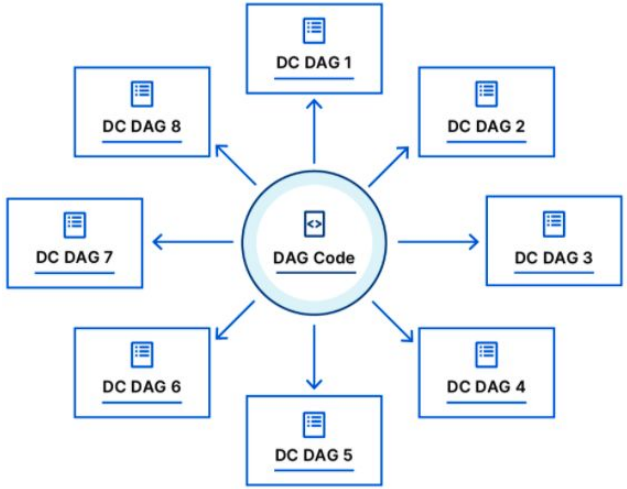


Phase 1 - machines are powered on, boots our custom Linux kernel

Phase 2 - newly provisioned machines are enabled to receive production traffic

CLOUDFLARE

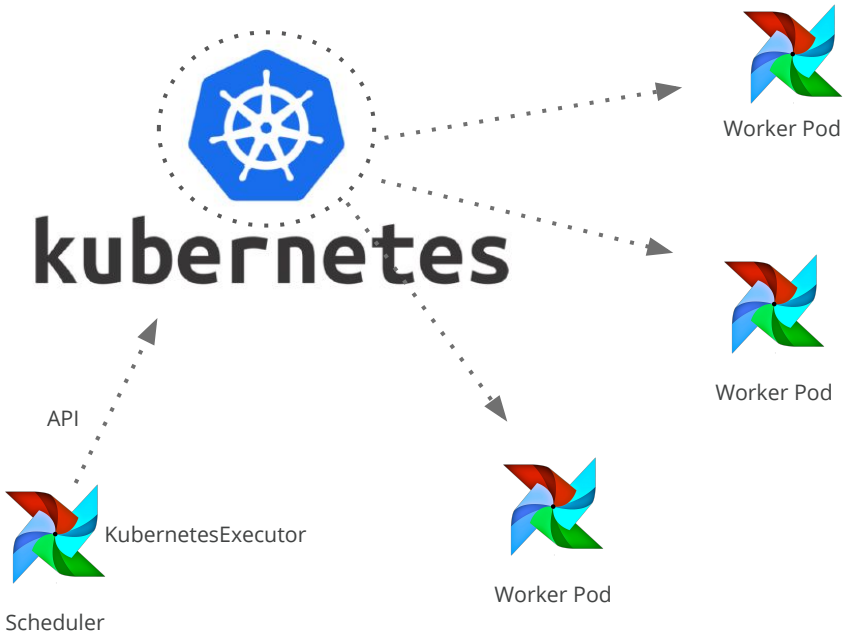
Creating DAGs that scale



Generating DAGs for each new data center instantly, without writing a single line of code.

CLOUDFLARE

Executing DAGs at scale

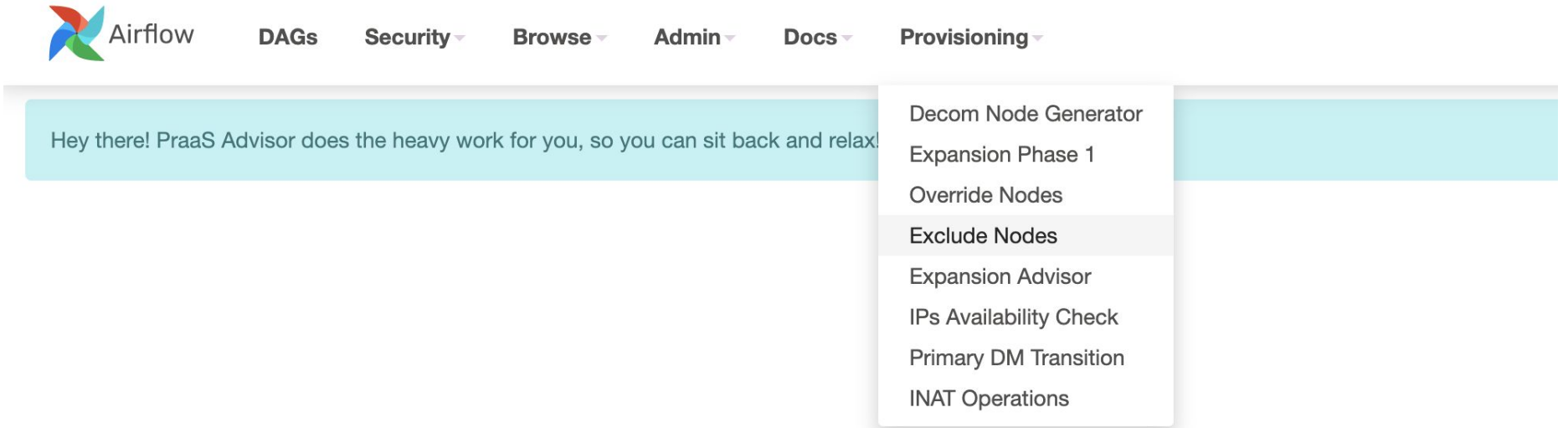


KubernetesExecutor - creates a new worker pod for every task instance that needs to be executed

The worker pod gets killed on completion of the task.

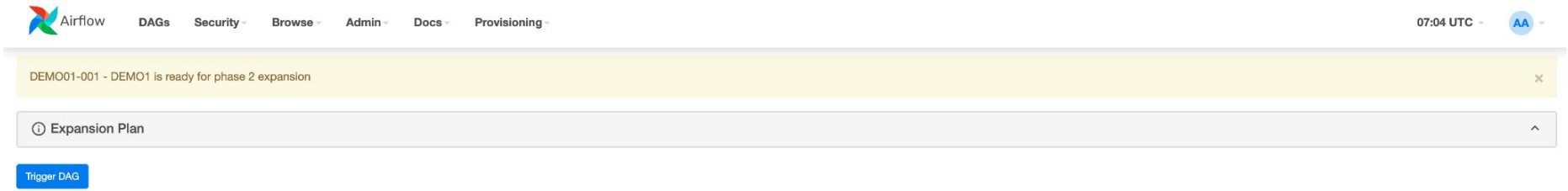
CLOUDFLARE

Improving the Usability and User experience



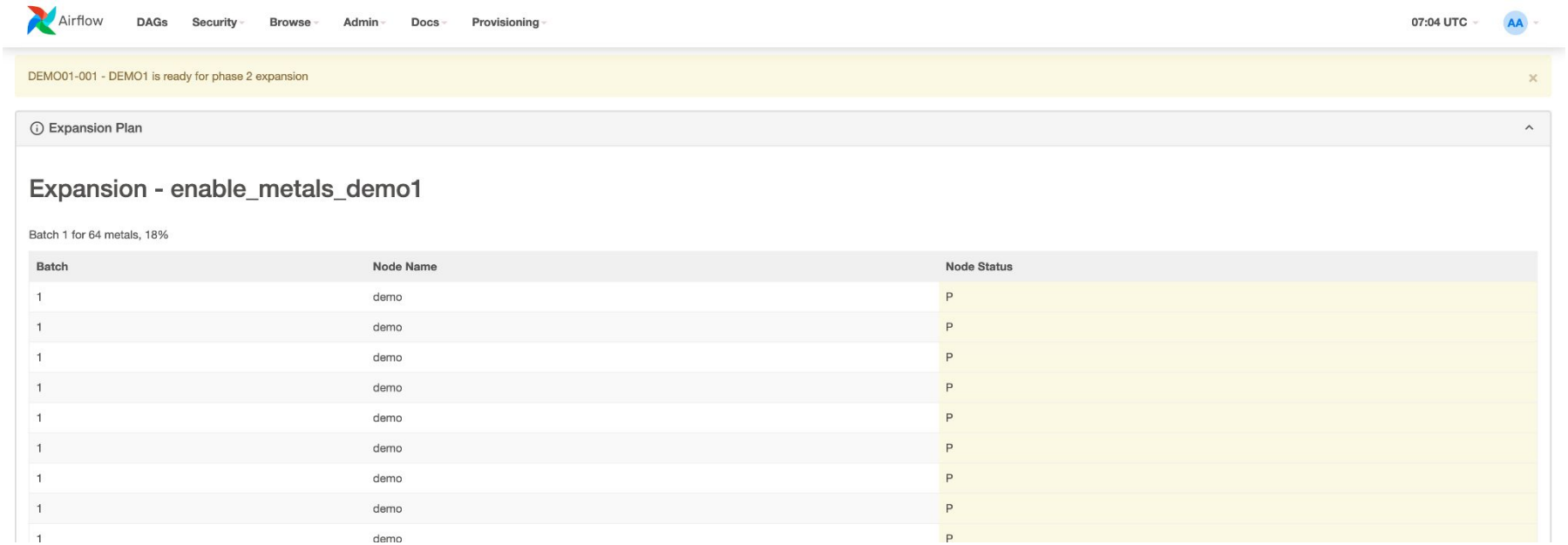
CLOUDFLARE

Improving the Usability and User experience



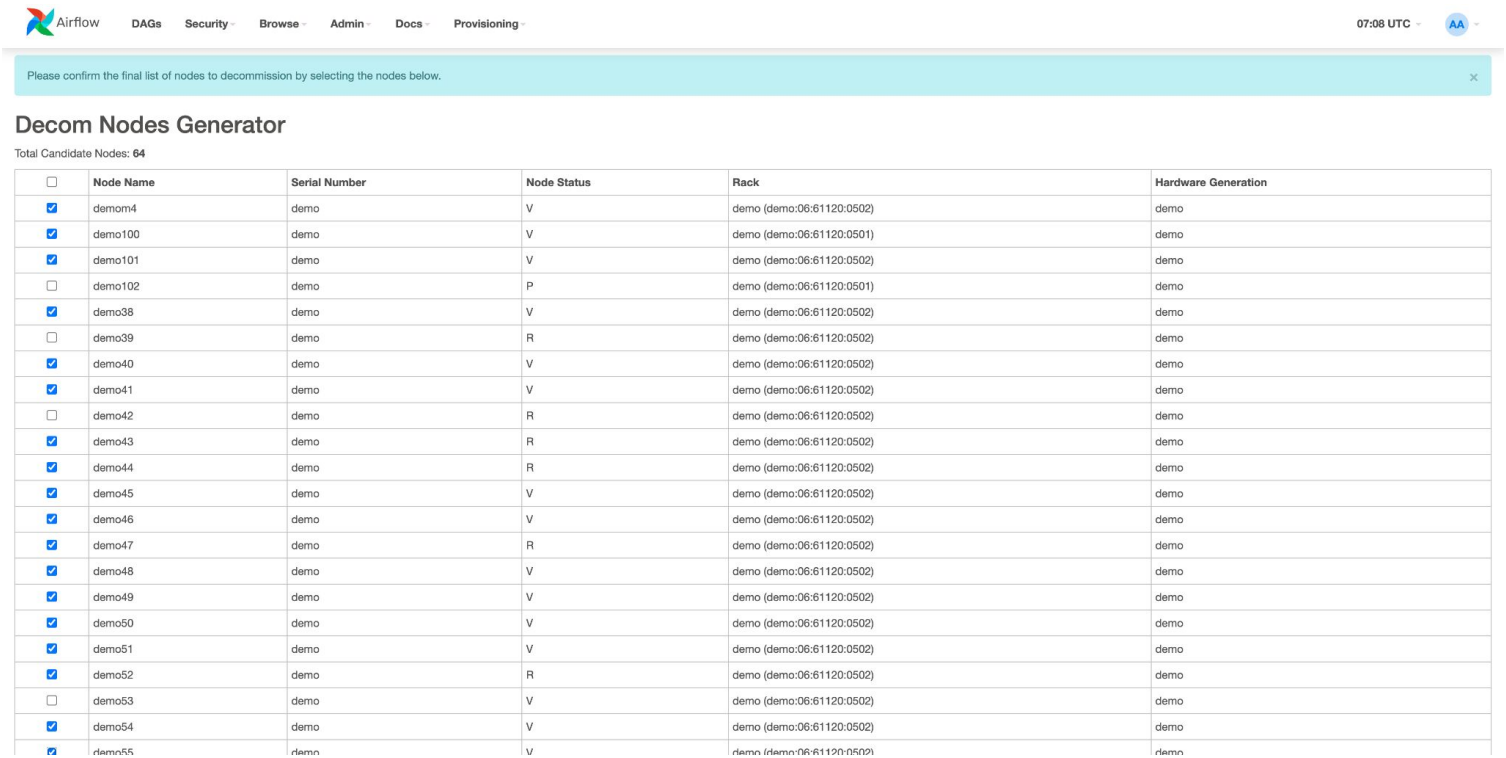
CLOUDFLARE

Improving the Usability and User experience



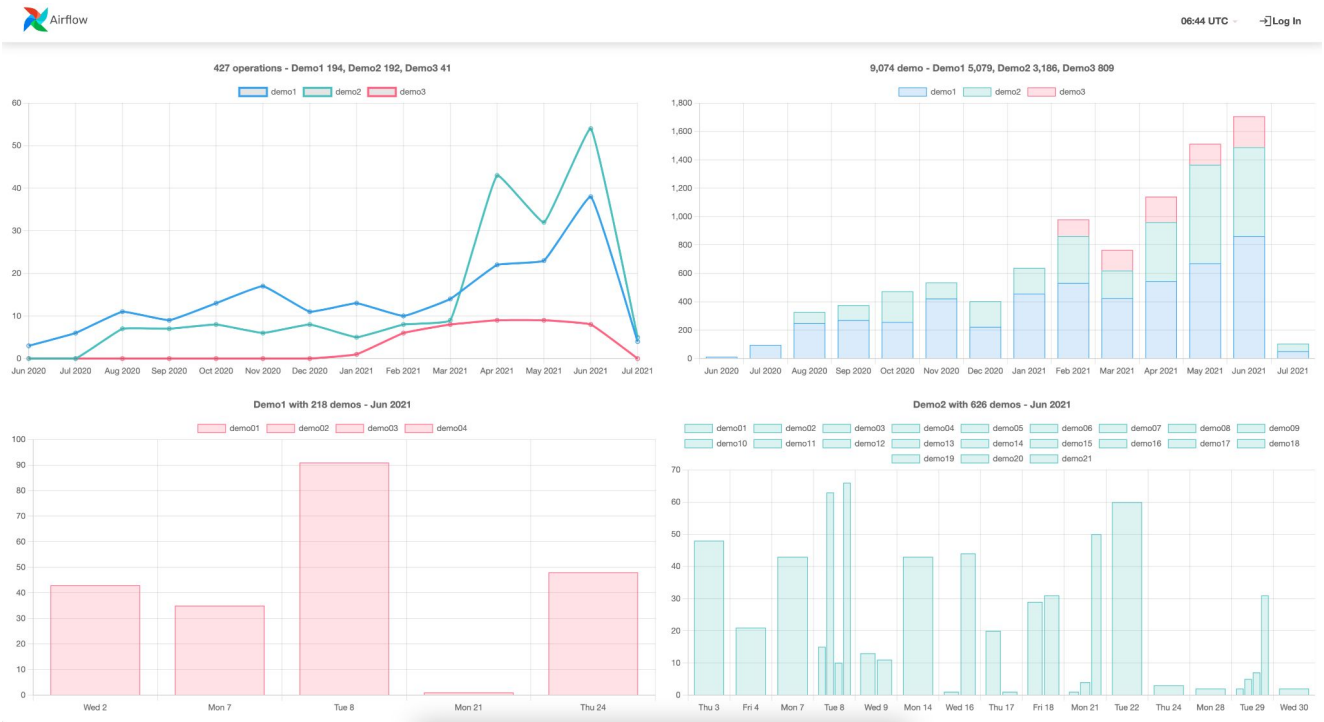
CLOUDFLARE

Improving the Usability and User experience



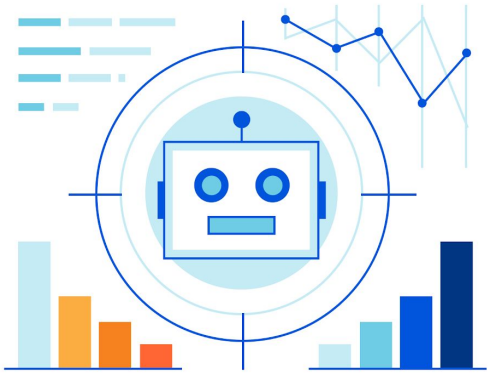
CLOUDFLARE

Custom dashboards for better insights



CLOUDFLARE

Ultimate Goal: Autonomous Provision-as-a-Service



For expansions, our ultimate goal is a fully autonomous system that monitors whether new servers have been racked in our edge data centers — and automatically triggers expansions — with no human intervention.

CLOUDFLARE

Thank You!