Provision-as-a-Service: Automating data center operations with Airflow at Cloudflare
Cloudflare is an intelligent, integrated global cloud network that delivers security, performance, and reliability for all your Internet infrastructure, people and connected devices.

CLOUDFLARE’S MISSION:

Help build a better Internet
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’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’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
What is Provisioning?
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.
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.
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.
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.
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.
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.

```python
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')
```
Custom Operators to integrate with other systems

- SaltStack
- Jira
- Google Chat
- PagerDuty
- Thanos / Prometheus
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.

```python
verify_node_dns = builder.wrap_class(DNSSensor)(
    task_id='verify_node_dns',
    zone='domain',
    nodes_from='{{ to_json(run_ctxglobals.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
)```
Accepting inputs and responding to human interventions
Custom forms for accepting user inputs

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Center</td>
<td>Please choose a data center. Please select a data center.</td>
</tr>
<tr>
<td>Jira Ticket</td>
<td>Please provide the DCA ticket id for this operation.</td>
</tr>
<tr>
<td>Nodes</td>
<td>Override with this space-separated list of nodes for this operation.</td>
</tr>
<tr>
<td>Operation</td>
<td>expansion</td>
</tr>
</tbody>
</table>

Save FI
Solving complex workflows with Branching and Multi-DAGs
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
Creating DAGs that scale

Generating DAGs for each new data center instantly, without writing a single line of code.
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.
Improving the Usability and User experience

Hey there! PraaS Advisor does the heavy work for you, so you can sit back and relax!
Improving the Usability and User experience
## Improving the Usability and User Experience

### Expansion - enable_metals_demo1

<table>
<thead>
<tr>
<th>Batch</th>
<th>Node Name</th>
<th>Node Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>demo</td>
<td>P</td>
</tr>
<tr>
<td>1</td>
<td>demo</td>
<td>P</td>
</tr>
<tr>
<td>1</td>
<td>demo</td>
<td>P</td>
</tr>
<tr>
<td>1</td>
<td>demo</td>
<td>P</td>
</tr>
<tr>
<td>1</td>
<td>demo</td>
<td>P</td>
</tr>
<tr>
<td>1</td>
<td>demo</td>
<td>P</td>
</tr>
<tr>
<td>1</td>
<td>demo</td>
<td>P</td>
</tr>
<tr>
<td>1</td>
<td>demo</td>
<td>P</td>
</tr>
</tbody>
</table>
### Improving the Usability and User Experience

#### Decom Nodes Generator

<table>
<thead>
<tr>
<th>Node Name</th>
<th>Serial Number</th>
<th>Node Status</th>
<th>Rack</th>
<th>Hardware Generation</th>
</tr>
</thead>
<tbody>
<tr>
<td>dem004</td>
<td>demo</td>
<td>V</td>
<td>demo (demo 06/8130/0120)</td>
<td>demo</td>
</tr>
<tr>
<td>dem0100</td>
<td>demo</td>
<td>V</td>
<td>demo (demo 06/8130/0110)</td>
<td>demo</td>
</tr>
<tr>
<td>dem0101</td>
<td>demo</td>
<td>V</td>
<td>demo (demo 06/8130/0100)</td>
<td>demo</td>
</tr>
<tr>
<td>dem0102</td>
<td>demo</td>
<td>P</td>
<td>demo (demo 06/8130/0100)</td>
<td>demo</td>
</tr>
<tr>
<td>dem038</td>
<td>demo</td>
<td>V</td>
<td>demo (demo 06/8130/0100)</td>
<td>demo</td>
</tr>
<tr>
<td>dem089</td>
<td>demo</td>
<td>R</td>
<td>demo (demo 06/8130/0100)</td>
<td>demo</td>
</tr>
<tr>
<td>dem040</td>
<td>demo</td>
<td>V</td>
<td>demo (demo 06/8130/0100)</td>
<td>demo</td>
</tr>
<tr>
<td>dem041</td>
<td>demo</td>
<td>V</td>
<td>demo (demo 06/8130/0100)</td>
<td>demo</td>
</tr>
<tr>
<td>dem042</td>
<td>demo</td>
<td>R</td>
<td>demo (demo 06/8130/0100)</td>
<td>demo</td>
</tr>
<tr>
<td>dem043</td>
<td>demo</td>
<td>R</td>
<td>demo (demo 06/8130/0100)</td>
<td>demo</td>
</tr>
<tr>
<td>dem044</td>
<td>demo</td>
<td>R</td>
<td>demo (demo 06/8130/0100)</td>
<td>demo</td>
</tr>
<tr>
<td>dem045</td>
<td>demo</td>
<td>V</td>
<td>demo (demo 06/8130/0100)</td>
<td>demo</td>
</tr>
<tr>
<td>dem046</td>
<td>demo</td>
<td>V</td>
<td>demo (demo 06/8130/0100)</td>
<td>demo</td>
</tr>
<tr>
<td>dem057</td>
<td>demo</td>
<td>R</td>
<td>demo (demo 06/8130/0100)</td>
<td>demo</td>
</tr>
<tr>
<td>dem048</td>
<td>demo</td>
<td>V</td>
<td>demo (demo 06/8130/0100)</td>
<td>demo</td>
</tr>
<tr>
<td>dem049</td>
<td>demo</td>
<td>V</td>
<td>demo (demo 06/8130/0100)</td>
<td>demo</td>
</tr>
<tr>
<td>dem050</td>
<td>demo</td>
<td>V</td>
<td>demo (demo 06/8130/0100)</td>
<td>demo</td>
</tr>
<tr>
<td>dem061</td>
<td>demo</td>
<td>V</td>
<td>demo (demo 06/8130/0100)</td>
<td>demo</td>
</tr>
<tr>
<td>dem062</td>
<td>demo</td>
<td>R</td>
<td>demo (demo 06/8130/0100)</td>
<td>demo</td>
</tr>
<tr>
<td>dem063</td>
<td>demo</td>
<td>V</td>
<td>demo (demo 06/8130/0100)</td>
<td>demo</td>
</tr>
<tr>
<td>dem064</td>
<td>demo</td>
<td>V</td>
<td>demo (demo 06/8130/0100)</td>
<td>demo</td>
</tr>
</tbody>
</table>
Custom dashboards for better insights
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.
Like our network, Cloudflare continues to rapidly grow. If working at a rapidly expanding, globally diverse company interests you, we’re hiring for scores of positions in the Infrastructure Engineering team.