PREFECT

PREFECT ASSOCIATE

CERTIFICATION

PREFECT

Introductions

Introductions

PREFECT ASSOCIATE
CERTIFICATION

- Off mic: emoji responses and threads in Slack
- On mic:
  - Name
  - Pronouns
  - Favorite animal 🐷🐍

PREFECT

Overview

Prefect

PREFECT ASSOCIATE
CERTIFICATION

Prefect provides tools for working with complex systems

so you can stop wondering about your workflows 🛑🤔

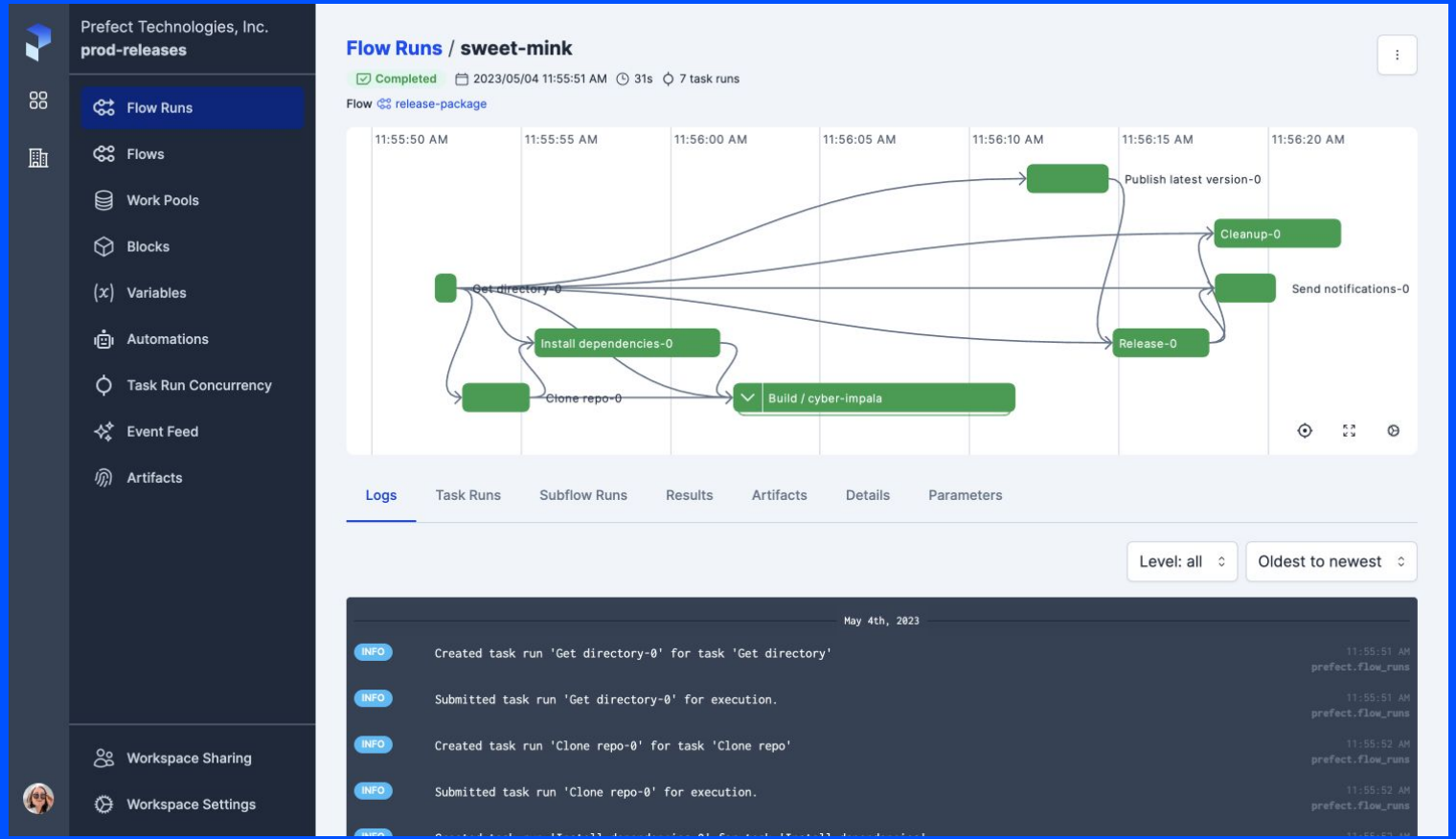
If you give an engineer a job...

PREFECT ASSOCIATE
CERTIFICATION

- Could you just move this data and clean it a bit?
- Could you set up logging?
- Could you do it every night?
- Could you make it retry if it fails?
- Could you send me a message when it succeeds?
- Could you visualize the dependencies?
- Could you add caching?
- Could you add collaborators to run ad hoc - who don't code?

Prefect makes it easy to orchestrate and observe Python code

PREFECT ASSOCIATE CERTIFICATION



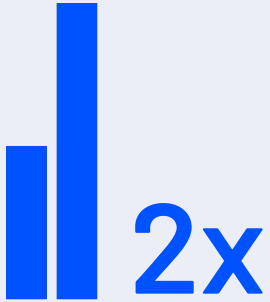
Prefect helps data teams be **more efficient and effective**

Prefect helps teams:

Develop Faster	Reduce Failures	Increase Visibility
How:		
• Intuitive Python-based library • Caching • Integrations • World-class support	• Clear & maintainable code • Test convenience • Easy async • Automatic retries	• UI • Notifications • Event feed • Collaboration

With **extraordinary impacts.**

of users reported *much* faster development cycles



average development productivity boost

reduction in pipeline errors

SOURCE: Prefect community survey, November 2021.

PREFECT

Goals

Goals

PREFECT ASSOCIATE
CERTIFICATION

1. Competence with Prefect 2
2. Connect with each other
3. Have fun! 🎉

PREFECT

Norms

Norms

PREFECT ASSOCIATE
CERTIFICATION

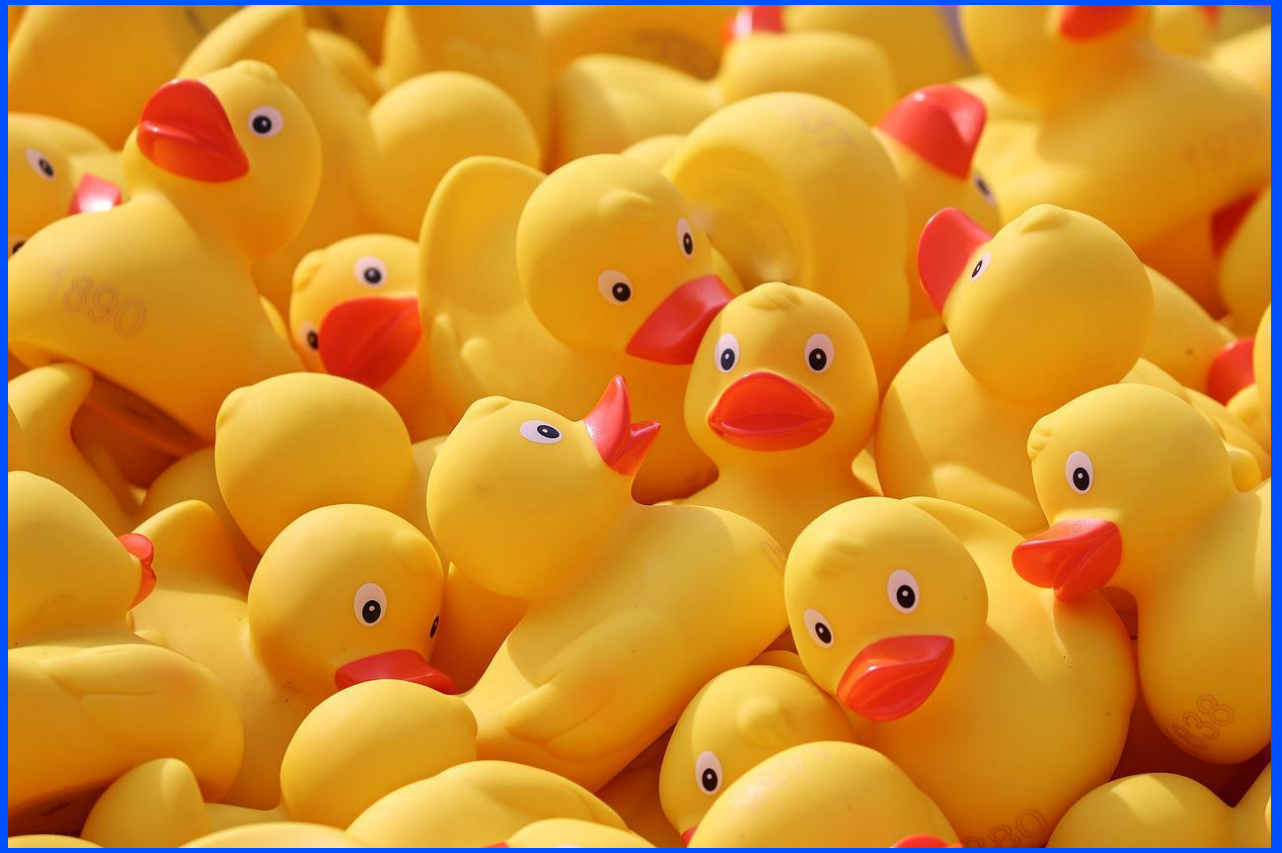
Zoom

- Camera on
- Mute unless asking a question
- Use hand raise to ask a question

Slack

- Use threads
- Emoji responses 🙂

Welcome to the land of the ducks

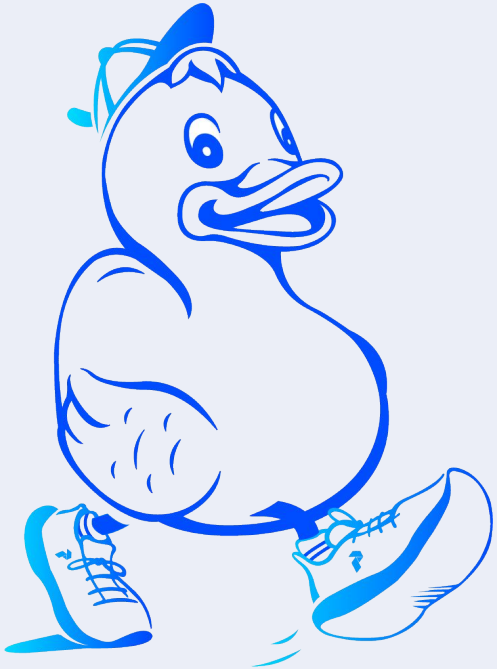


PREFECT ASSOCIATE
CERTIFICATION

Meet Minerva

Minerva does data things for the Quackme Co. consulting firm.

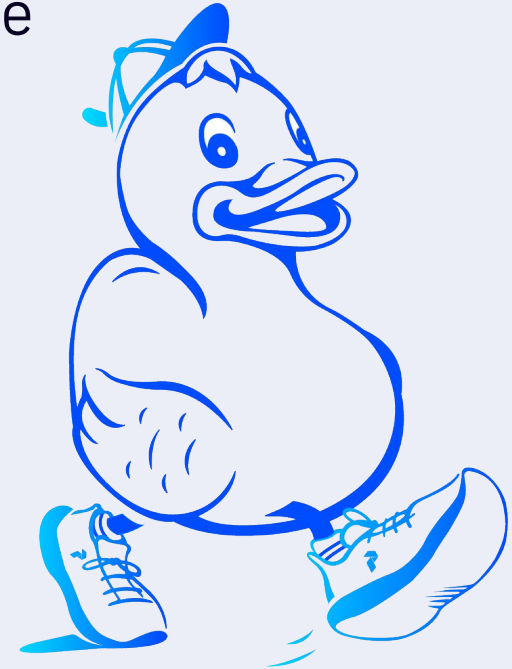
- Weather forecasts 🌦️
- Cat & dog facts 🐱🐶
- Stock data 📈



Your charge

Help Minerva build data pipelines that are

- Reliable
- Resilient
- Observable
- Scheduled
- Easy to understand



PREFECT

MODULE

101 - Prefect basics

101 Agenda

PREFECT ASSOCIATE
CERTIFICATION

- From Python function to Prefect flow
- Tasks
- Docs

Initial project

PREFECT ASSOCIATE
CERTIFICATION

Help Minerva get some weather data. 

open-meteo.com/en

Fetch data: Basic Python function

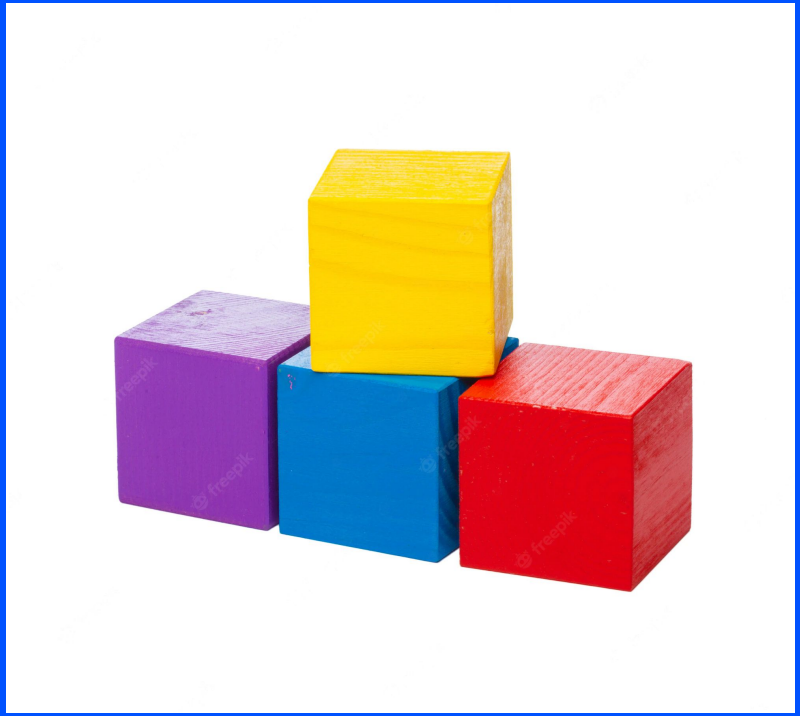
PREFECT ASSOCIATE
CERTIFICATION

```python
import httpx

def fetch_weather(lat: float, lon: float):
    base_url = "https://api.open-meteo.com/v1/forecast/"
    weather = httpx.get(
        base_url,
        params=dict(latitude=lat, longitude=lon, hourly="temperature_2m"),
    )
    most_recent_temp = float(weather.json()["hourly"]["temperature_2m"][0])
    print(f"Most recent temp C: {most_recent_temp} degrees")
    return most_recent_temp


if __name__ == "__main__":
    fetch_weather(38.9, -77.0)
```

Flows

PREFECT ASSOCIATE
CERTIFICATION

- Add a Prefect @flow
- Most basic Prefect object
- All you need to start

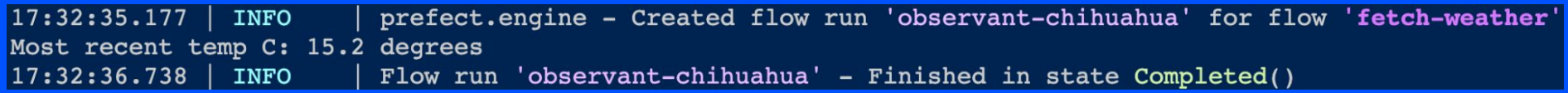


Make it a flow

PREFECT ASSOCIATE
CERTIFICATION

```python
from prefect import flow

@flow
def fetch_weather(lat: float, lon: float):
    base_url = "https://api.open-meteo.com/v1/forecast/"
    weather = httpx.get(
        base_url,
        params=dict(latitude=lat, longitude=lon, hourly="temperature_2m"),
    )
    most_recent_temp = float(weather.json()["hourly"]["temperature_2m"][0])
    print(f"Most recent temp C: {most_recent_temp} degrees")
    return most_recent_temp
```

Flow results



Concepts

PREFECT ASSOCIATE
CERTIFICATION

Flows

Tasks

Pipeline

PREFECT ASSOCIATE
CERTIFICATION

1. Fetch weather data and return it ✅
2. Save data to csv and return success message 🙂
3. Pipeline to call the other two 📞

Fetch data function

PREFECT ASSOCIATE
CERTIFICATION

```python
def fetch_weather(lat: float, lon: float):
    base_url = "https://api.open-meteo.com/v1/forecast/"
    weather = httpx.get(
        base_url,
        params=dict(latitude=lat, longitude=lon, hourly="temperature_2m"),
    )
    most_recent_temp = float(weather.json()["hourly"]["temperature_2m"][0])
    return most_recent_temp
```

Save data function

PREFECT ASSOCIATE
CERTIFICATION

```python
def save_weather(temp: float):
    with open("weather.csv", "w+") as w:
        w.write(str(temp))
    return "Successfully wrote temp"
```

Pipeline (assembly) function

PREFECT ASSOCIATE
CERTIFICATION

```python
def pipeline(lat: float, lon: float):
    temp = fetch_weather(lat, lon)
    result = save_weather(temp)
    return result
```

Tasks

PREFECT ASSOCIATE
CERTIFICATION

Turn the first two functions into *tasks*

Turn into tasks

PREFECT ASSOCIATE
CERTIFICATION

```python
from prefect import flow, task

@task
def fetch_weather(lat: float, lon: float):
    base_url = "https://api.open-meteo.com/v1/forecast/"
    weather = httpx.get(
        base_url,
        params=dict(latitude=lat, longitude=lon, hourly="temperature_2m"),
    )
    most_recent_temp = float(weather.json()["hourly"]["temperature_2m"][0])
    return most_recent_temp
```

Turn into tasks

PREFECT ASSOCIATE
CERTIFICATION

```python
@task
def save_weather(temp: float):
    with open("weather.csv", "w+") as w:
        w.write(str(temp))
    return "Successfully wrote temp"
```

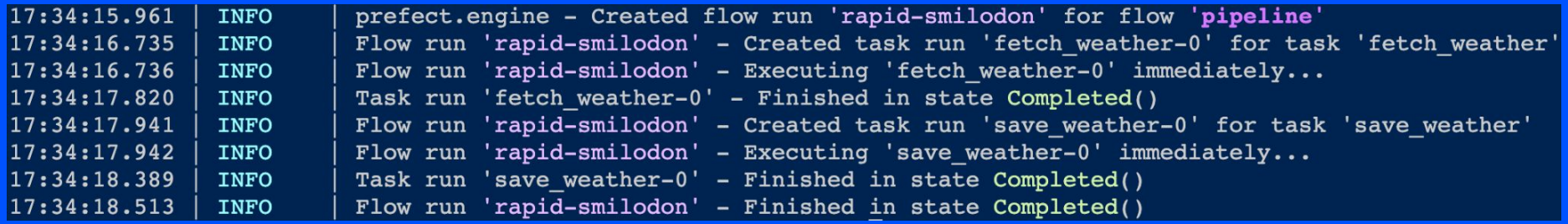
Pipeline function flow

PREFECT ASSOCIATE
CERTIFICATION

Pass the result of one task to another inside a flow

```python
@flow
def pipeline(lat: float, lon: float):
    temp = fetch_weather(lat, lon)
    result = save_weather(temp)
    return result
```

Logs from run



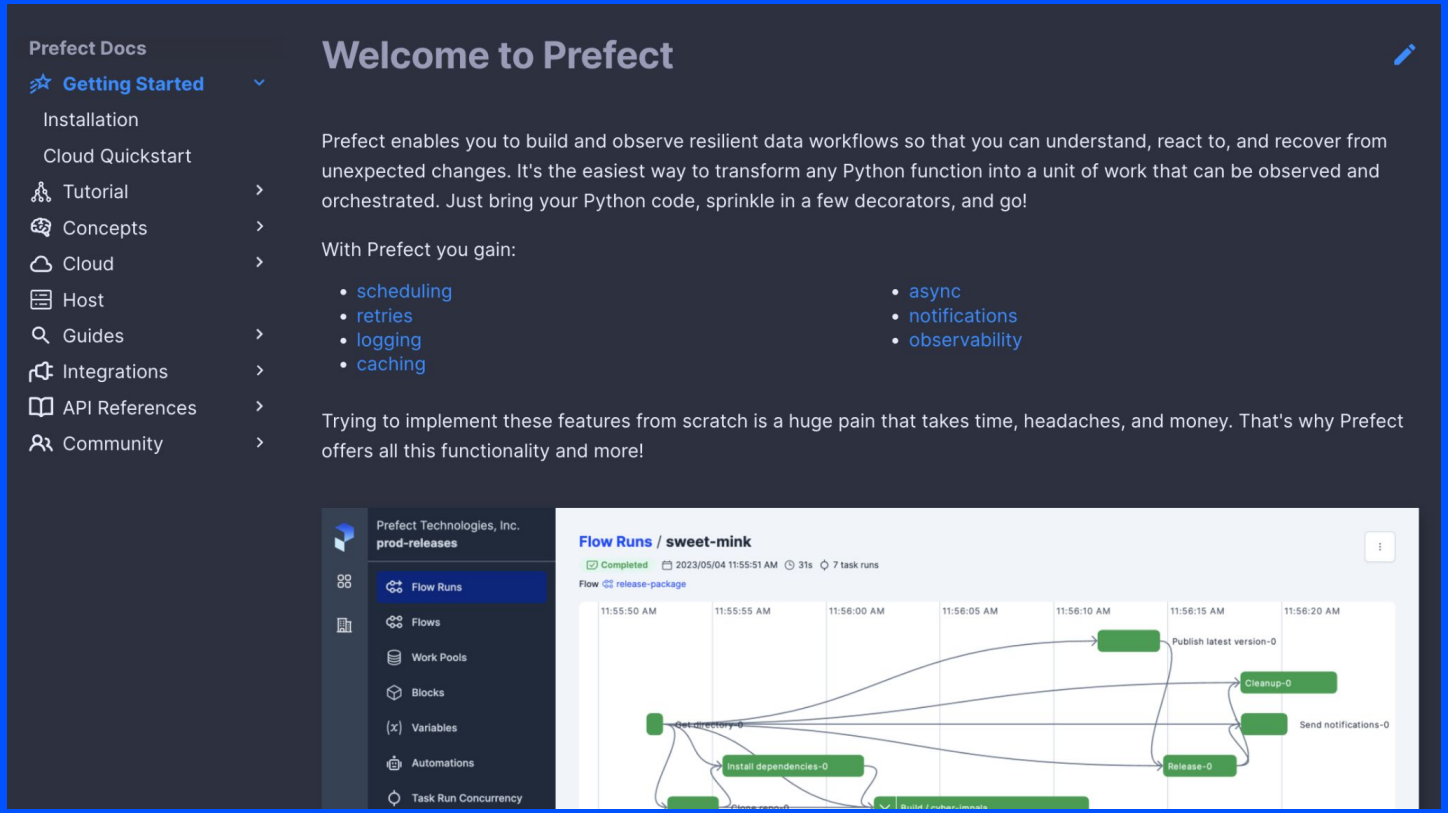
Tasks dos and don’ts

PREFECT ASSOCIATE
CERTIFICATION

- ✅ Do call tasks within a flow
- ⛔ Don’t call tasks from other tasks
- ✅ Do keep tasks small

Docs

PREFECT ASSOCIATE
CERTIFICATION



Docs - *docs.prefect.io*

PREFECT ASSOCIATE
CERTIFICATION

Use the docs

Prefect details

PREFECT ASSOCIATE
CERTIFICATION

prefect version

```
Version:              2.10.8
API version:          0.8.4
Python version:       3.9.12
Git commit:           79093235
Built:                Mon, May 8, 2023 12:23 PM
OS/Arch:              darwin/arm64
Profile:              default
Server type:          cloud
```

Resources

PREFECT ASSOCIATE
CERTIFICATION

- Prefect Community Slack:

  https://www.prefect.io/slack/

- Prefect Discourse:

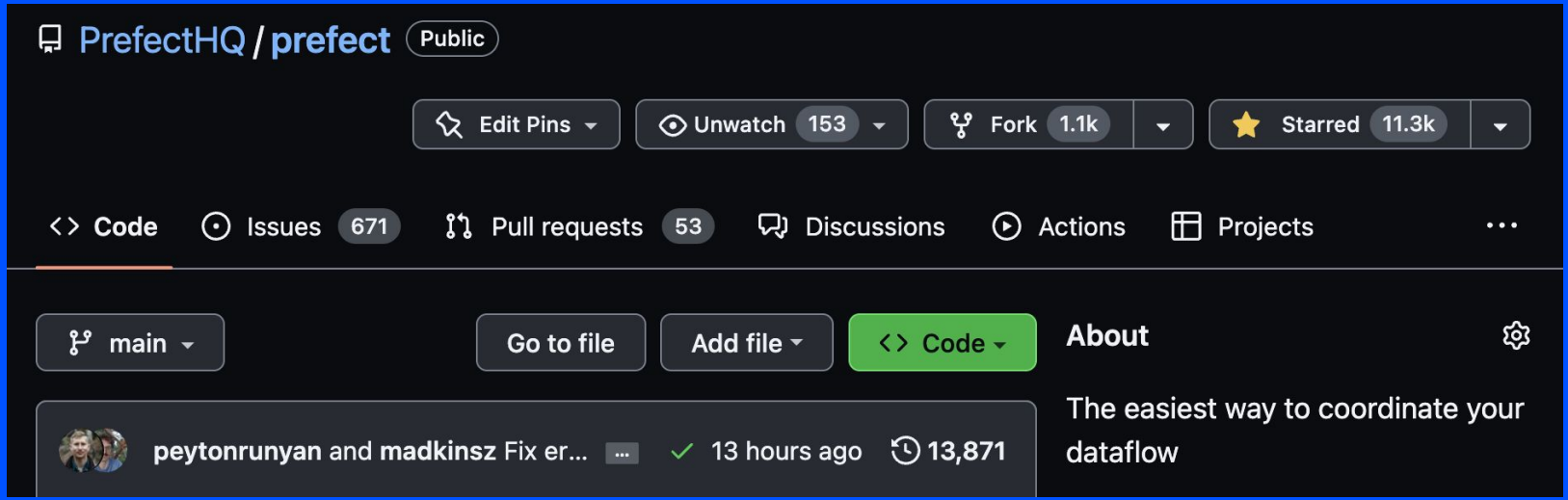
  https://discourse.prefect.io/

Prefect codebase

PREFECT ASSOCIATE
CERTIFICATION

https://github.com/PrefectHQ/prefect

Give it a ⭐



101 Recap

You've seen how to get started with Prefect!

- From Python function to Prefect flow
- Create modular tasks that can be called **from inside** a flow
- Docs are your friends
- GitHub Codebase, Slack, Discourse are helpful

PREFECT

Lab 101

101 Lab

PREFECT ASSOCIATE CERTIFICATION

Use Open-Meteo API

- Write flow code that fetches other weather metrics
- Make at least 3 tasks in the flow
- Example: windspeed for the last hour:

```
weather.json()["hourly"]["windspeed_10m"][0]
```

- Docs: open-meteo.com/en/docs

PREFECT

MODULE

102 - Intro to orchestration

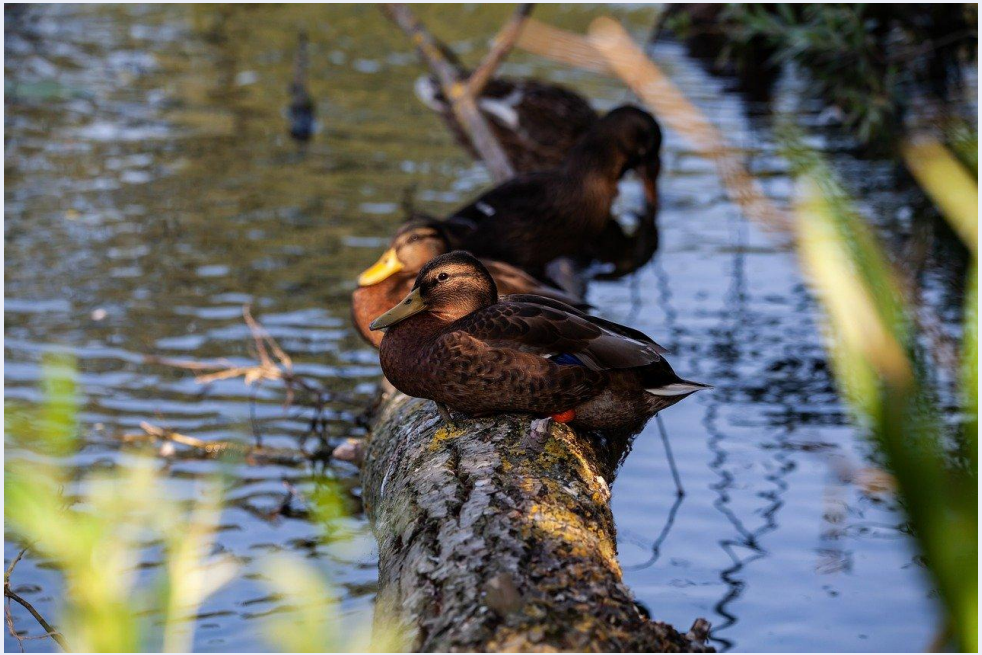
102 Agenda

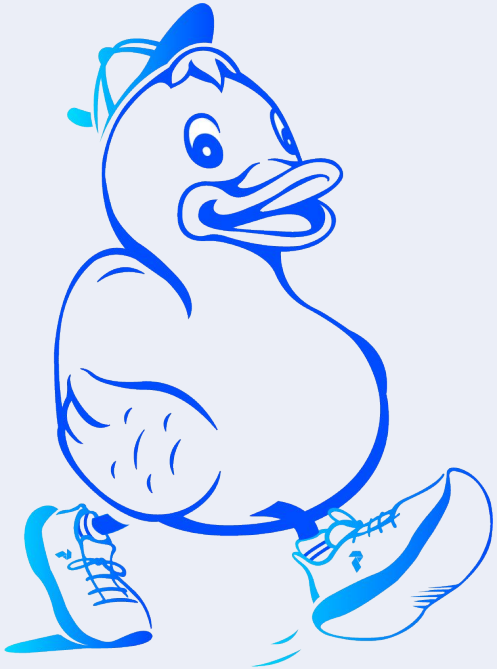
PREFECT ASSOCIATE
CERTIFICATION

- Logging
- Retries for tasks and flows
- API server
- Results
- Subflows
- Learn about cats

Logging

Ducks on logs:





log_prints

Log the print statements

```
@flow(log_prints=True)
```

Logging

Create custom logs with `get_run_logger`

```python
from prefect import flow, get_run_logger

@flow(name="log-example-flow")
def log_it():
    logger = get_run_logger()
    logger.info("INFO level log message.")
    logger.debug("You only see this message if the logging level is set to DEBUG. 🙂")

if __name__ == "__main__":
    log_it()
```

Logging

PREFECT ASSOCIATE CERTIFICATION

Output with DEBUG logging level:

```
14:27:11.137 | DEBUG   | prefect.profiles - Using profile 'local'
14:27:11.674 | DEBUG   | prefect.client - Using ephemeral application with database at sqlite
+aiosqlite:////Users/jeffhale/.prefect/prefect.db
14:27:11.727 | INFO    | prefect.engine - Created flow run 'heavy-nightingale' for flow 'log-
example-flow'
14:27:11.727 | DEBUG   | Flow run 'heavy-nightingale' - Starting 'ConcurrentTaskRunner'; subm
itted tasks will be run concurrently...
14:27:11.728 | DEBUG   | prefect.task_runner.concurrent - Starting task runner...
14:27:11.729 | DEBUG   | prefect.client - Using ephemeral application with database at sqlite
+aiosqlite:////Users/jeffhale/.prefect/prefect.db
14:27:11.799 | DEBUG   | Flow run 'heavy-nightingale' - Executing flow 'log-example-flow' for
 flow run 'heavy-nightingale'...
14:27:11.799 | DEBUG   | Flow run 'heavy-nightingale' - Beginning execution...
14:27:11.799 | INFO    | Flow run 'heavy-nightingale' - INFO level log message.
14:27:11.800 | DEBUG   | Flow run 'heavy-nightingale' - You only see this message if the logg
ing level is set to DEBUG. 🙂
14:27:11.818 | DEBUG   | prefect.task_runner.concurrent - Shutting down task runner...
14:27:11.818 | INFO    | Flow run 'heavy-nightingale' - Finished in state Completed()
```

Logging

Output with INFO logging level:

```
14:24:55.950 | INFO    | prefect.engine - Created flow run 'macho-sturgeon' for flow 'log-exa
mple-flow'
14:24:56.022 | INFO    | Flow run 'macho-sturgeon' - INFO level log message.
14:24:56.041 | INFO    | Flow run 'macho-sturgeon' - Finished in state Completed()
```

Cat fact data

Minerva's a smart duck. She made an app to share cat facts.

Unfortunately, her app uses a cat-fact API that can be a bit buggy.

Retries

PREFECT ASSOCIATE
CERTIFICATION

Specify in task or a flow decorator

```
@task(retries=2)
```

```
@flow(retries=3)
```

Flow retries for an unreliable API

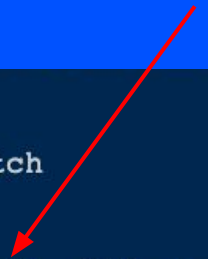
PREFECT ASSOCIATE
CERTIFICATION

```python
@flow(retries=4)
def fetch():
    cat_fact = httpx.get("https://f3-vyx5c2hfpq-ue.a.run.app/")
    if cat_fact.status_code >= 400:
        raise Exception()
    print(cat_fact.text)
```

Fail then automatic retry

PREFECT ASSOCIATE
CERTIFICATION

```
line 135, in capture_worker_thread_and_result
    result = __fn(*args, **kwargs)
  File "/Users/jeffhale/Desktop/prefect/demos/pacc/retry-flow.py", line 9, in fetch
    raise Exception()
Exception
22:23:21.040 | INFO    | Flow run 'mysterious-agouti' - Received non-final state 'AwaitingRetry' w
hen proposing final state 'Failed' and will attempt to run again...
Cats have the largest eyes of any mammal.
22:23:22.534 | INFO    | Flow run 'mysterious-agouti' - Finished in state Completed()
```



Retry delays

PREFECT ASSOCIATE
CERTIFICATION

Retry delay

PREFECT ASSOCIATE
CERTIFICATION

Specify in task or flow decorator

```
@task(retries=2, retry_delay_seconds=0.1)
```

Task retries with delay

PREFECT ASSOCIATE
CERTIFICATION

```python
@task(retries=4, retry_delay_seconds=0.1)
def fetch_cat_fact():
    cat_fact = httpx.get("https://f3-vyx5c2hfpq-ue.a.run.app/")
    if cat_fact.status_code >= 400:
        raise Exception()
    print(cat_fact.text)
```

Fail then automatic retry with delay

PREFECT ASSOCIATE
CERTIFICATION

```
Exception
22:30:35.032 | INFO    | Task run 'fetch_cat_fact-0' - Received non-final state 'AwaitingRetry' wh
en proposing final state 'Failed' and will attempt to run again...
Blue-eyed, pure white cats are frequently deaf.
22:30:36.856 | INFO    | Task run 'fetch_cat_fact-0' - Finished in state Completed()
22:30:36.874 | INFO    | Flow run 'ivory-finch' - Finished in state Completed('All states complete
d.')
```

PREFECT

Prefect server

Start a Prefect server locally

Open another terminal window

Run:

```
prefect server start
```

The Prefect CLI

PREFECT ASSOCIATE
CERTIFICATION

Start commands with *prefect*

--help is always available

prefect --help

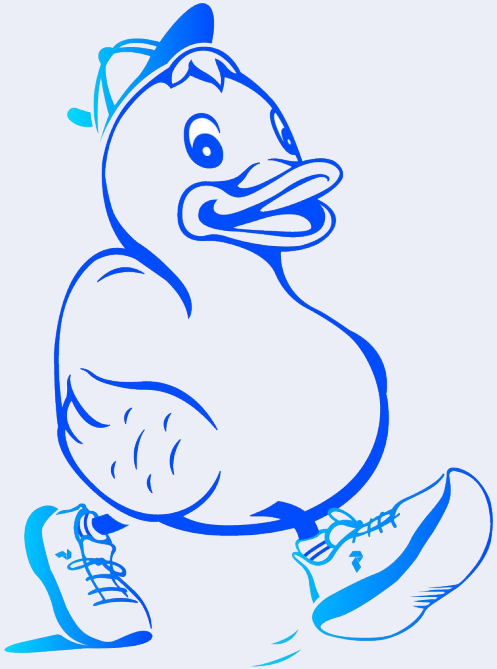
PREFECT ASSOCIATE
CERTIFICATION

```
Options
--version  -v            Display the current version.
--profile  -p     TEXT   Select a profile for this this CLI run. [default: None]
--help                   Show this message and exit.

Commands
agent                    Commands for starting and interacting with agent processes.
block                    Commands for working with blocks.
cloud                    Commands for interacting with Prefect Cloud
concurrency-limit        Commands for managing task-level concurrency limits.
config                   Commands for interacting with Prefect settings.
deployment               Commands for working with deployments.
dev                      Commands for development.
flow                     Commands for interacting with flows.
flow-run                 Commands for interacting with flow runs.
kubernetes               Commands for working with Orion on Kubernetes.
orion                    Commands for interacting with backend services.
profile                  Commands for interacting with your Prefect profiles.
version                  Get the current Prefect version.
work-pool                Commands for working with work pools.
work-queue               Commands for working with work queues.
```

Let's get Visual 📊

PREFECT ASSOCIATE
CERTIFICATION

Fire up the Prefect Server



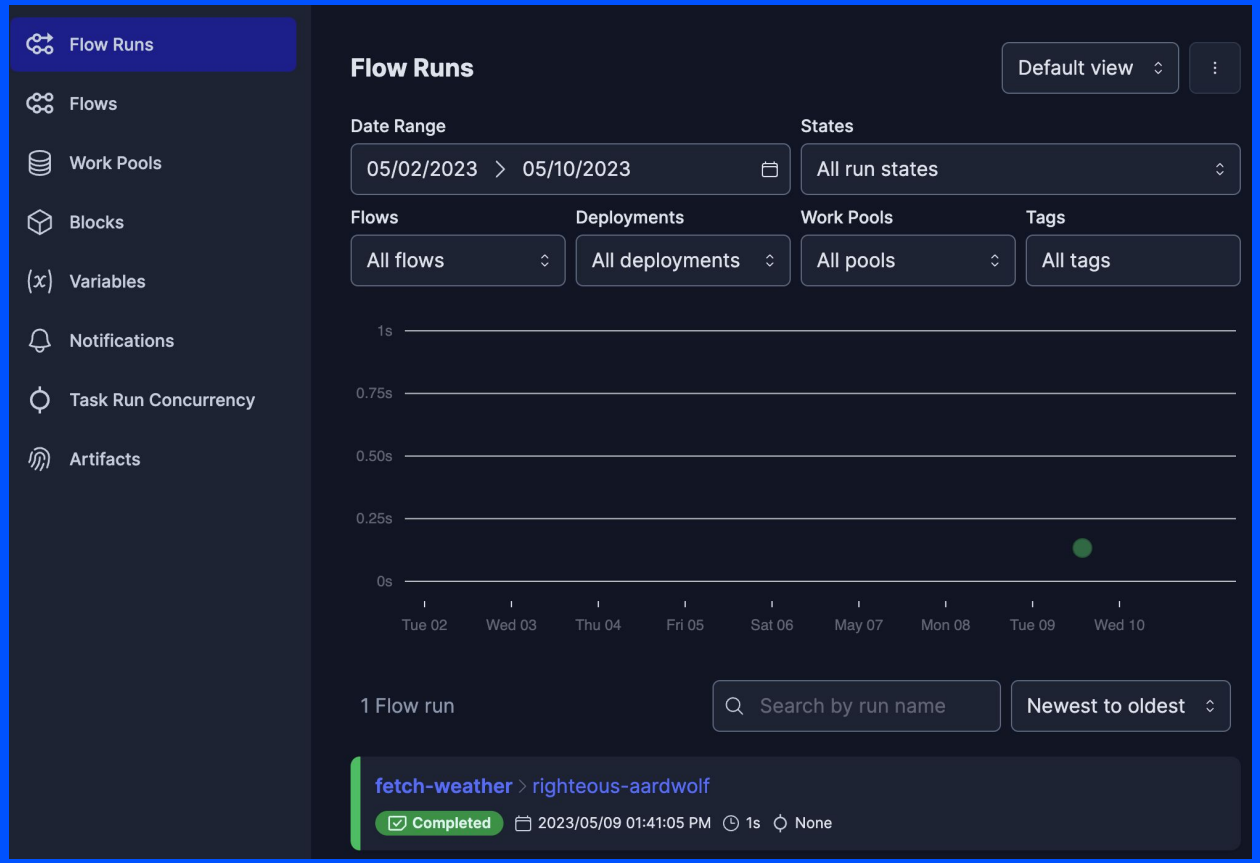
Start a Prefect server locally

PREFECT ASSOCIATE
CERTIFICATION

```
 ___ ___ ___ ___ ___ ___ _____
| _ \ _ \ __| __| __/ __|_   _|
|  _/   / _|| _|| _| (__  | |
|_| |_|_\___|_| |___\___| |_|

Configure Prefect to communicate with the server with:

    prefect config set PREFECT_API_URL=http://127.0.0.1:4200/api

View the API reference documentation at http://127.0.0.1:4200/docs

Check out the dashboard at http://127.0.0.1:4200
```

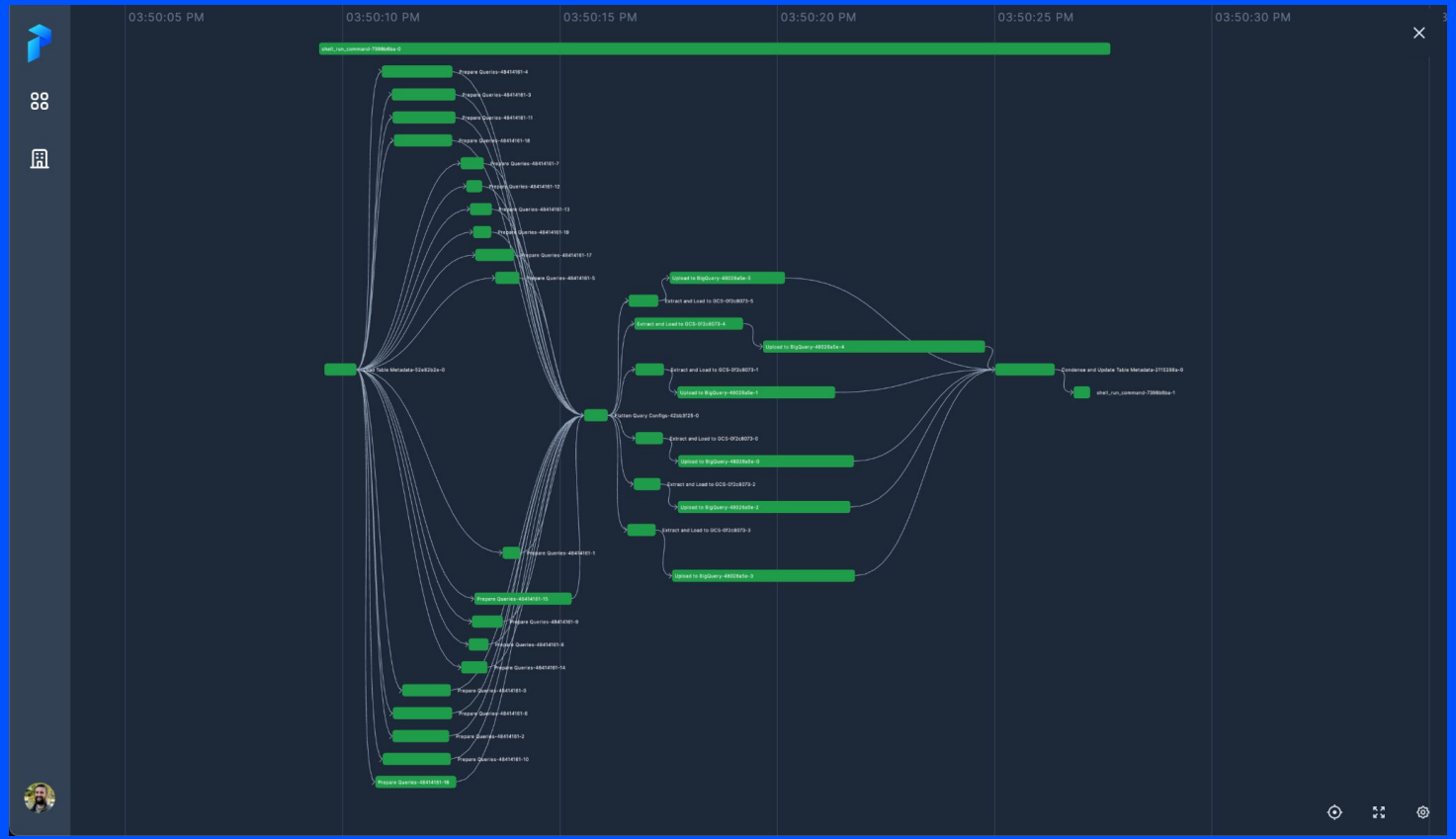
Prefect UI

PREFECT ASSOCIATE
CERTIFICATION



Prefect UI

PREFECT ASSOCIATE
CERTIFICATION



PREFECT

Results

Results

PREFECT ASSOCIATE
CERTIFICATION

The data returned by a flow or a task

By default, Prefect returns a result that is not persisted to disk.

Results

The data returned by a flow or a task

```
@task
def my_task():
    return 1
```

1 is the result

Results

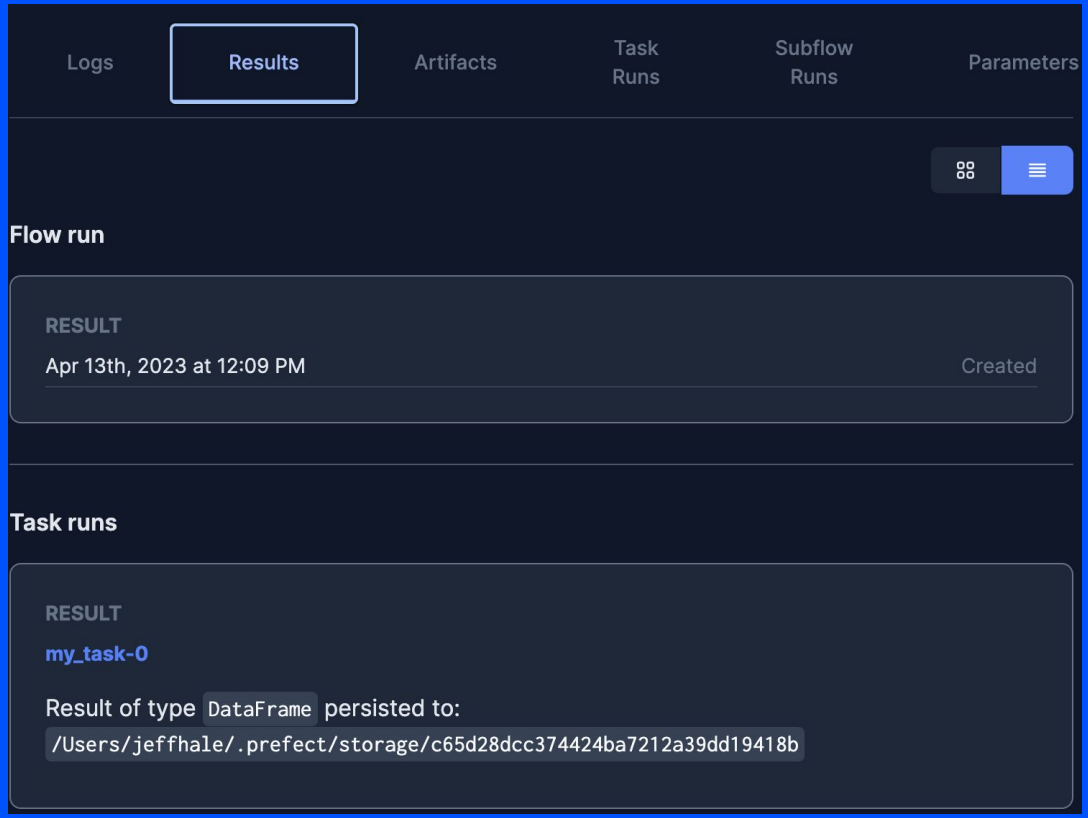
PREFECT ASSOCIATE
CERTIFICATION

```python
from prefect import flow, task
import pandas as pd

@task(persist_result=True)
def my_task():
    df = pd.DataFrame(dict(a=[2, 3], b=[4, 5]))
    return df

@flow()
def my_flow():
    res = my_task()

my_flow()
```

Results

PREFECT ASSOCIATE CERTIFICATION

Viewable in the UI



Results

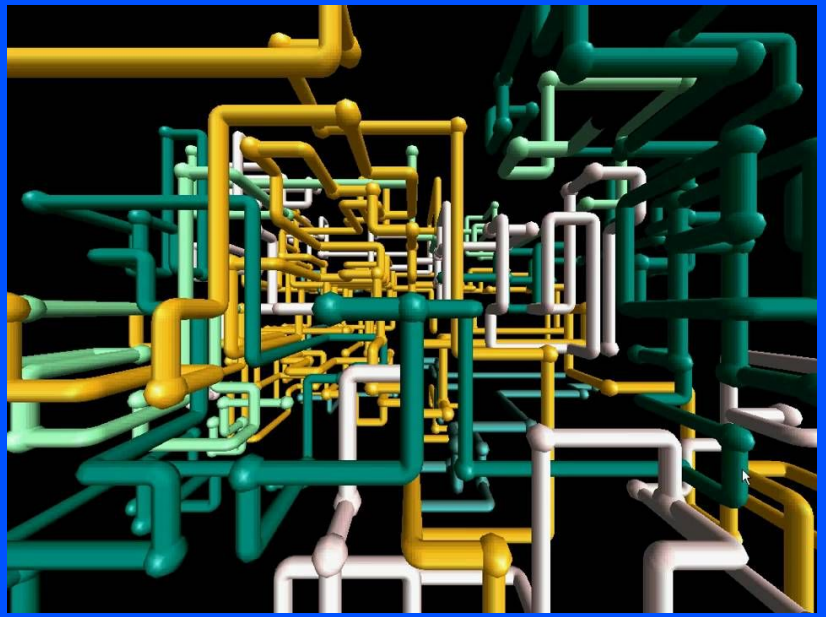
PREFECT ASSOCIATE
CERTIFICATION

The result in the *storage* folder

Subflows

PREFECT ASSOCIATE
CERTIFICATION

- A flow that calls another flow
- Useful for grouping related tasks



Subflows 🐈 + 🐩

PREFECT ASSOCIATE
CERTIFICATION

```python
@flow
def fetch_cat_fact():
    return httpx.get("https://catfact.ninja/fact?max_length=140").json()["fact"]

@flow
def fetch_dog_fact():
    return httpx.get(
        "https://dogapi.dog/api/v2/facts",
        headers={"accept": "application/json"},
    ).json()["data"][0]["attributes"]["body"]

@flow(log_prints=True)
def animal_facts():
    cat_fact = fetch_cat_fact()
    dog_fact = fetch_dog_fact()
    print(f"🐱: {cat_fact} \n🐶: {dog_fact}")
```

Subflows 🐈 + 🐩

PREFECT ASSOCIATE
CERTIFICATION

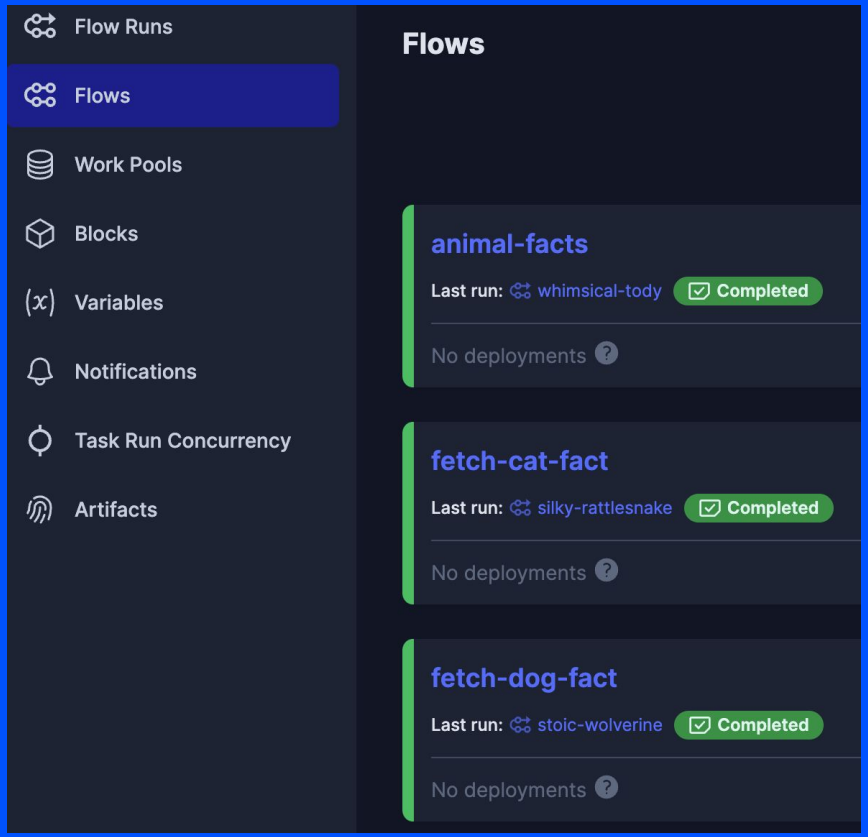
```
14:16:09.654 | INFO    | prefect.engine - Created flow run 'whimsical-tody' for flow 'animal-facts'
14:16:09.760 | INFO    | Flow run 'whimsical-tody' - Created subflow run 'silky-rattlesnake' for flow 'fetch-cat-fact'
14:16:09.988 | INFO    | Flow run 'silky-rattlesnake' - Finished in state Completed()
14:16:10.029 | INFO    | Flow run 'whimsical-tody' - Created subflow run 'stoic-wolverine' for flow 'fetch-dog-fact'
14:16:10.612 | INFO    | Flow run 'stoic-wolverine' - Finished in state Completed()
14:16:10.613 | INFO    | Flow run 'whimsical-tody' - 🐱: A cat lover is called an Ailurophilia (Greek: cat+lover).
🐶: Wow, check out those choppers! Puppies have 28 teeth and normal adult dogs have 42.
14:16:10.633 | INFO    | Flow run 'whimsical-tody' - Finished in state Completed('All states completed.')
```

Run over to the UI

PREFECT ASSOCIATE
CERTIFICATION

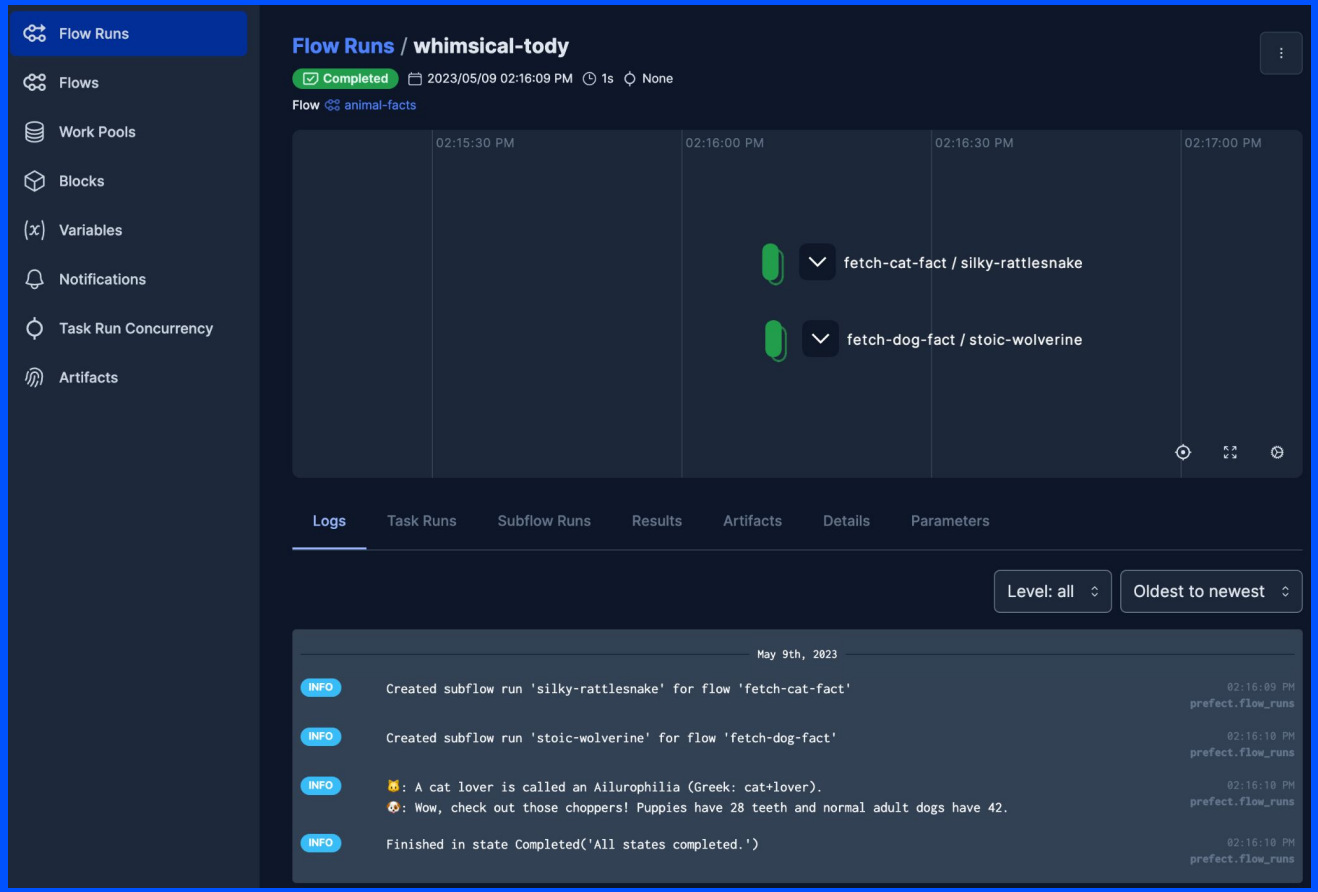
Subflows in UI

PREFECT ASSOCIATE
CERTIFICATION



Timeline view

PREFECT ASSOCIATE
CERTIFICATION



Logs in CLI

PREFECT ASSOCIATE
CERTIFICATION

```
                                    Feb 6th, 2023
INFO    Created subflow run 'fractal-hamster' for flow 'fetch-cat-fact'          10:05:15 PM
INFO    Created subflow run 'xanthic-nautilus' for flow 'fetch-dog-fact'         10:05:15 PM
INFO    🐱: Cats have the largest eyes of any mammal.                            10:05:16 PM
        🐶: A dog's nose is the equivalent of a human's fingerprints, as no two dogs have
        the same nose print.
INFO    Finished in state Completed('All states completed.')                     10:05:16 PM
```

102 Recap

PREFECT ASSOCIATE
CERTIFICATION

You've seen more of the power of Prefect tasks and flows.

- Logging
- Retries for tasks and flows
- API server
- Results
- Subflows

PREFECT

Lab 102

Lab 102

PREFECT ASSOCIATE
CERTIFICATION

- Make a new flow that grabs different weather data from open-meteo
- Add retries
- Add a subflow that transforms or aggregates the data

PREFECT

MODULE

103 - Cloud & Blocks

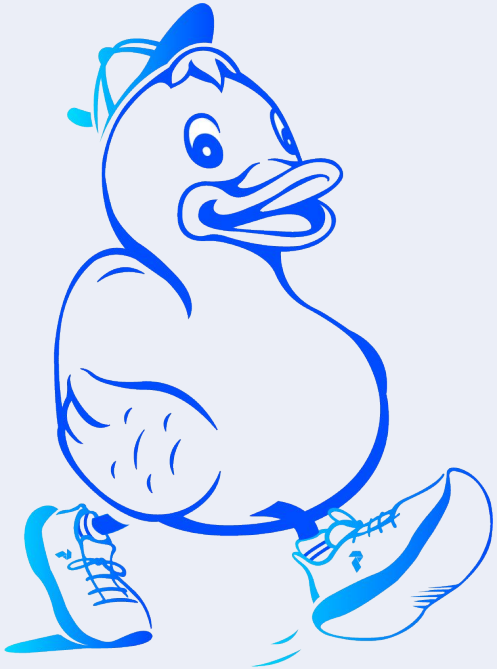
103 Agenda

PREFECT ASSOCIATE
CERTIFICATION

- Cloud
- Hybrid Model
- Intro to Blocks 🧱
- Automations 🗣️

To the cloud

Like other ducks, Minerva is into clouds.



Prefect Cloud

Go to app.prefect.cloud in browser

- Sign up or sign in

Prefect Cloud

PREFECT ASSOCIATE
CERTIFICATION

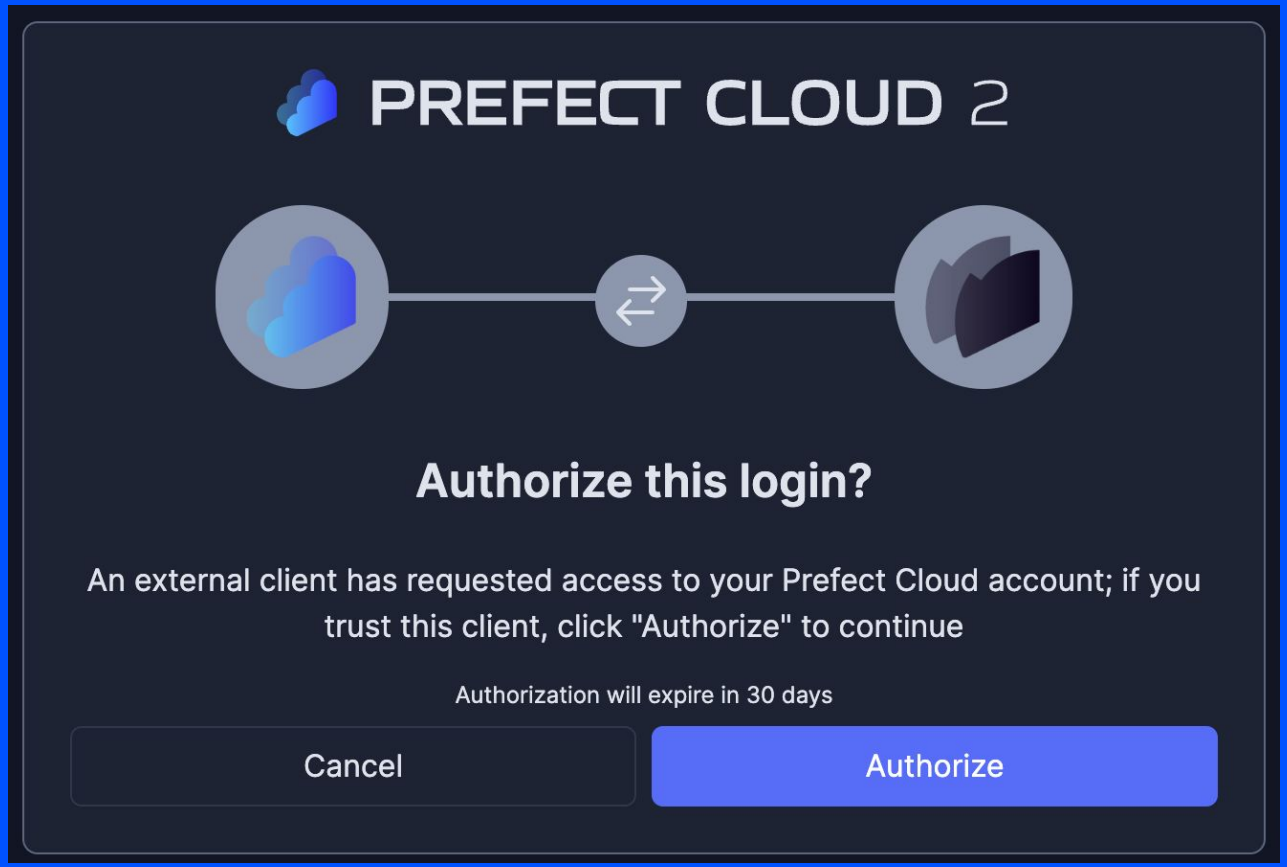
Authenticate CLI through browser or API key

Run: `prefect cloud login`

```
? How would you like to authenticate? [Use arrows to move; enter to select]
> Log in with a web browser
  Paste an API key
```

Prefect Cloud - log in with a web browser

PREFECT ASSOCIATE
CERTIFICATION



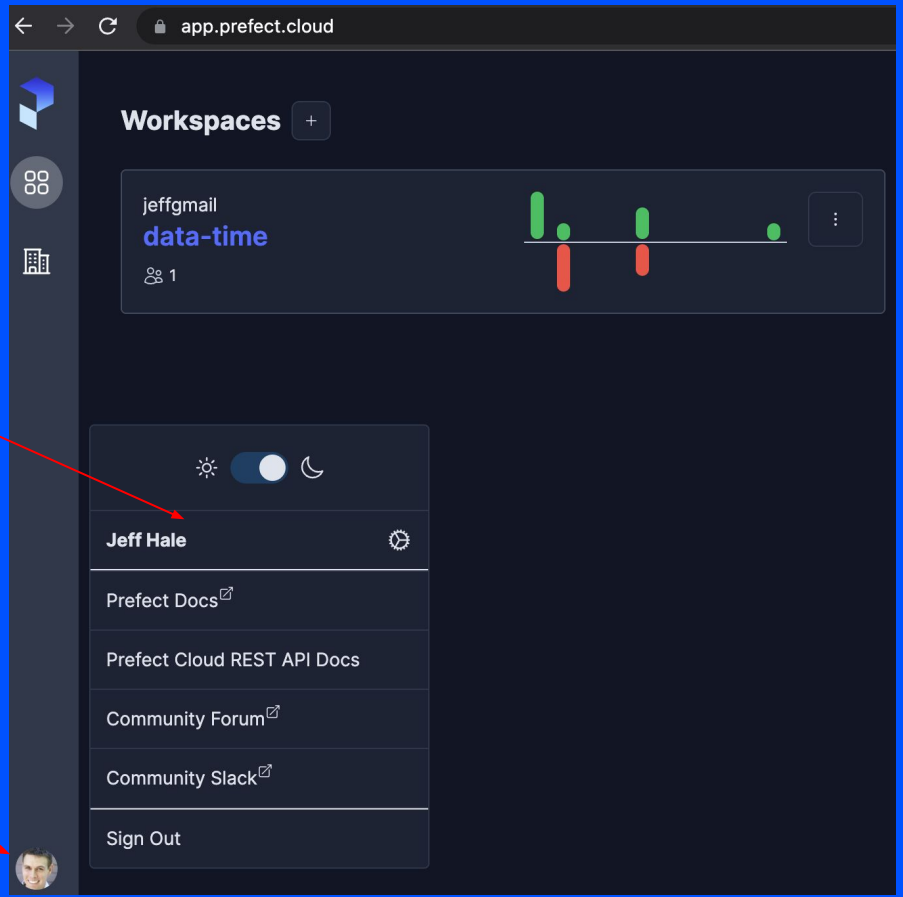
Prefect Cloud

Or paste an API key

Manually grab an API key from Prefect Cloud in the browser

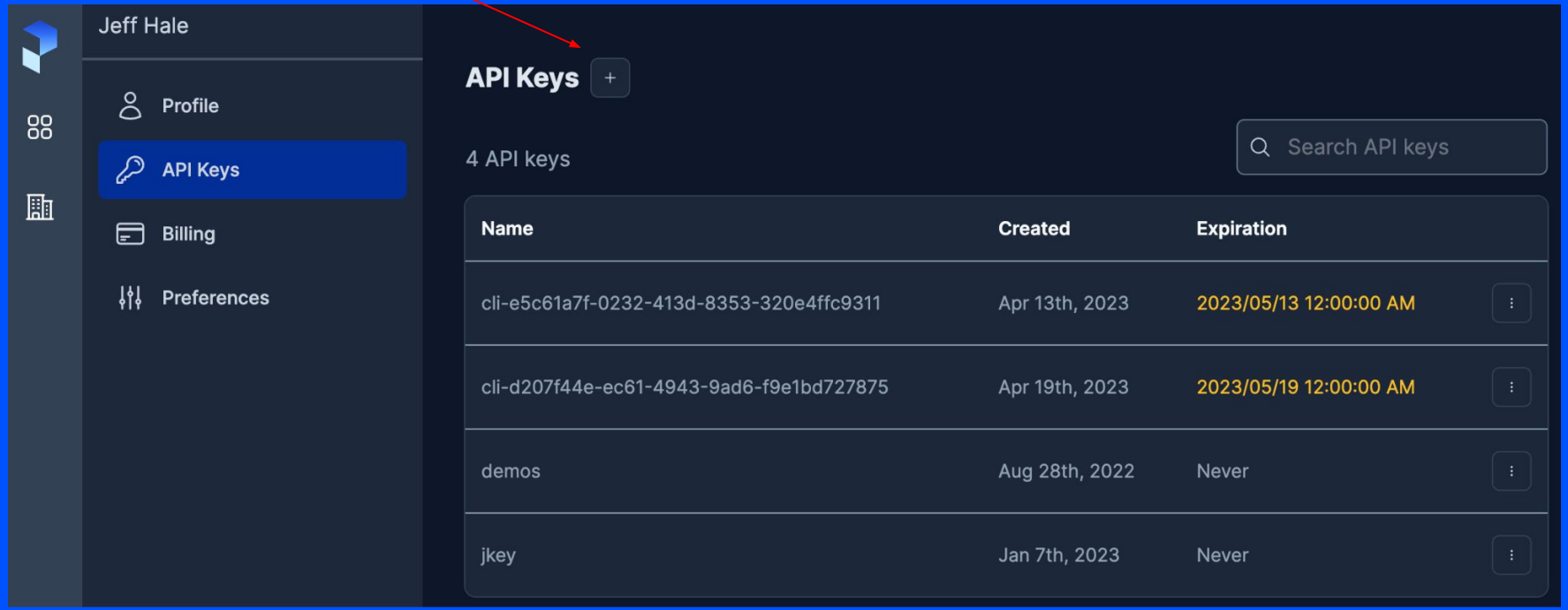
Prefect Cloud - API Key

PREFECT ASSOCIATE
CERTIFICATION



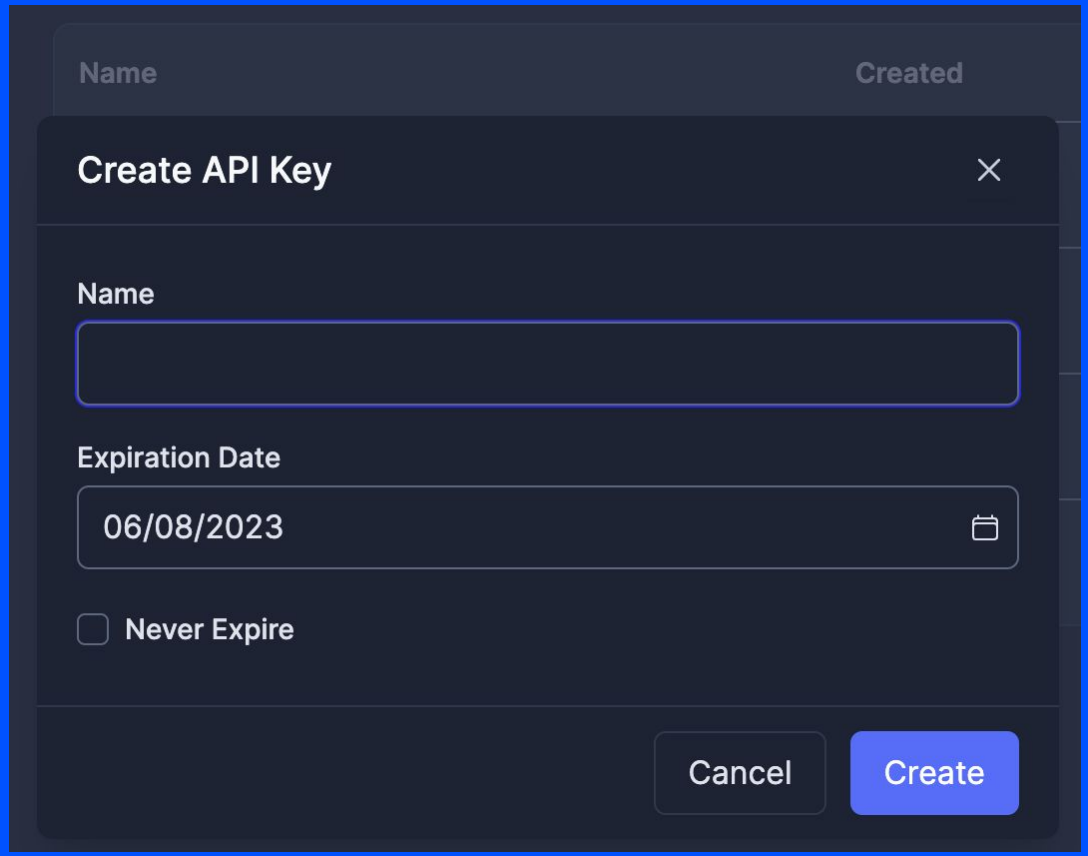
Prefect Cloud - API Key

PREFECT ASSOCIATE CERTIFICATION



Prefect Cloud - API Key

PREFECT ASSOCIATE
CERTIFICATION



PREFECT

Profiles

Prefect profiles - list

PREFECT ASSOCIATE
CERTIFICATION

```
prefect profile ls
```

```
Available Profiles:
* default
  local
  jeffmshale
  gh2
  prefect-more
```

Profiles

PREFECT ASSOCIATE
CERTIFICATION

Persist settings that you can switch between

Common for switching between

- Cloud and server
- Different Cloud workspaces

Profiles

PREFECT ASSOCIATE
CERTIFICATION

Create: `prefect profile create my_profile`

Inspect: `prefect profile inspect`

Select: `prefect profile use my_profile`

Profiles live in `~/.prefect/profiles.toml`

PREFECT

Prefect Cloud

Prefect Cloud

PREFECT ASSOCIATE
CERTIFICATION

- Server hosted by Prefect
- Workspaces
- Service Accounts
- RBAC
- SSO
- Automations
- Events

Prefect Cloud Workspaces

PREFECT ASSOCIATE
CERTIFICATION

- Paid plans can have multiple workspaces
- Each workspace is self-contained

Prefect Cloud - Personal

PREFECT ASSOCIATE
CERTIFICATION

- 3 free users (can buy more seats) - self service
- 1 free workspace
- 7 day flow run history

Prefect Cloud - Organization

PREFECT ASSOCIATE
CERTIFICATION

- Service accounts
- SSO
- RBAC
- 30 day flow run history
- 72 hour audit log

Prefect Cloud - Enterprise

PREFECT ASSOCIATE
CERTIFICATION

- Custom RBAC
- SSO SCIM (e.g. Okta)
- Custom most everything 🙂

Prefect Cloud

PREFECT ASSOCIATE
CERTIFICATION

Personal	Organization	Enterprise	Dedicated Cloud
Start with managed orchestration.	Prefect for scaling out with your team. Get a free 14 day trial.	Security, compliance and custom configurations.	Dedicated Instance of Prefect Cloud
Free Forever	Starts at **$450**/mo	**Get an estimate**	**Get in touch**
CREATE YOUR WORKSPACE	START YOUR FREE TRIAL	TALK TO SALES	TALK TO A PLATFORM ENGINEER
3 free Users	Read-Only Users	Enterprise support	No Resource Limits
1 free Workspace	Service Accounts + Enforced SSO	SCIM + Custom RBAC	Performance x Security
7 day history	30 day history	Custom configurations	Implementation and Extended SLA

Prefect Cloud - Default Roles (Org + Enterprise)

PREFECT ASSOCIATE
CERTIFICATION

Organization level

- Admin
- Member

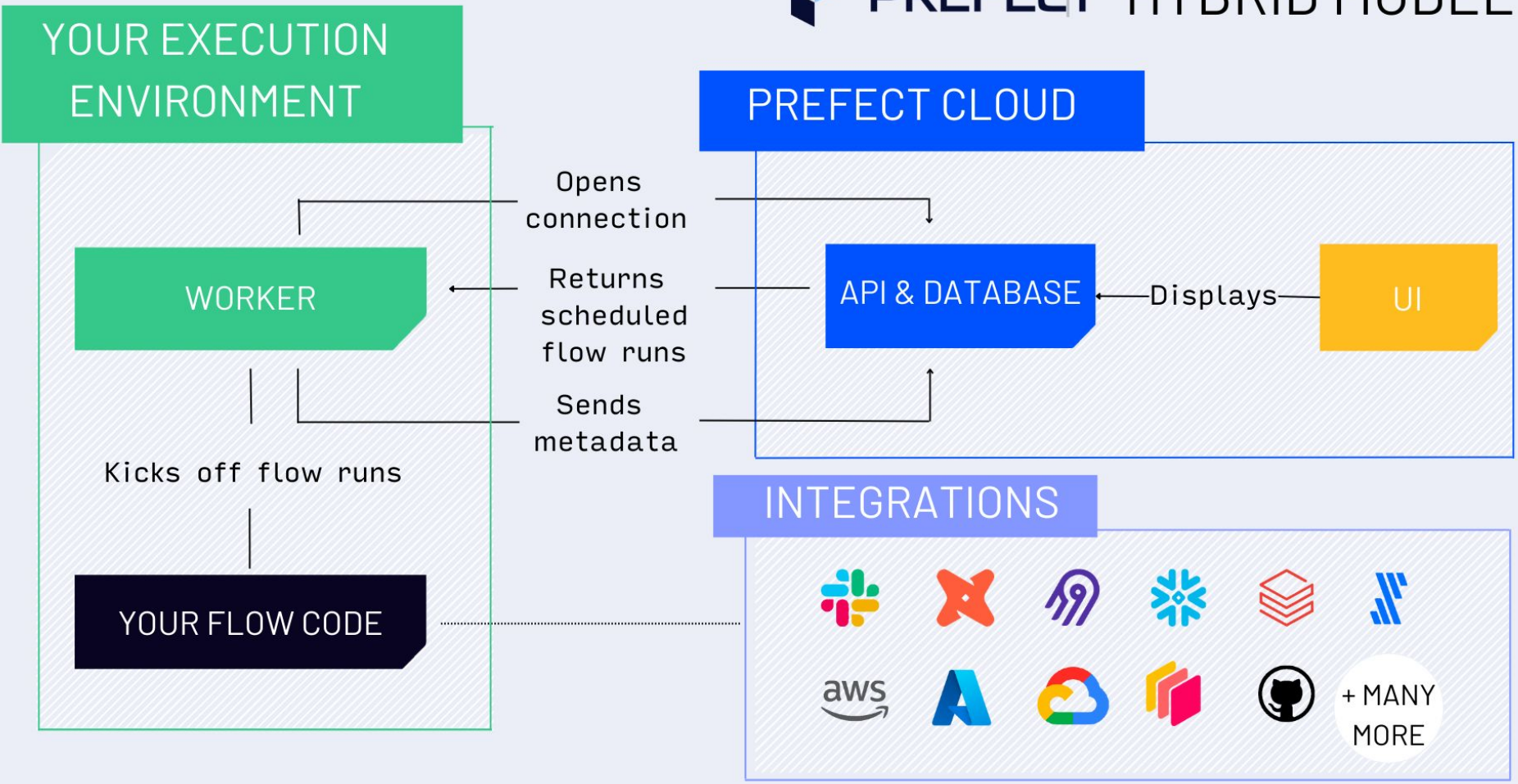
Workspace level

- Viewer
- Runner
- Developer
- Owner

PREFECT

Hybrid Model

PREFECT HYBRID MODEL



Hybrid model

- Your flow code runs on your infrastructure
- Prefect Cloud stores metadata and logs
- Data encrypted at rest
- Prefect Technologies, Inc. is SOC2 Type II compliant

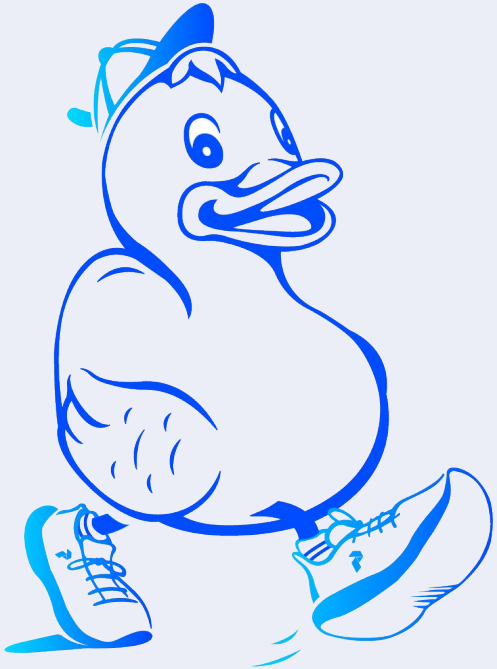
https://www.prefect.io/security

PREFECT

Blocks

Blocks

Blocks are a cool Prefect feature



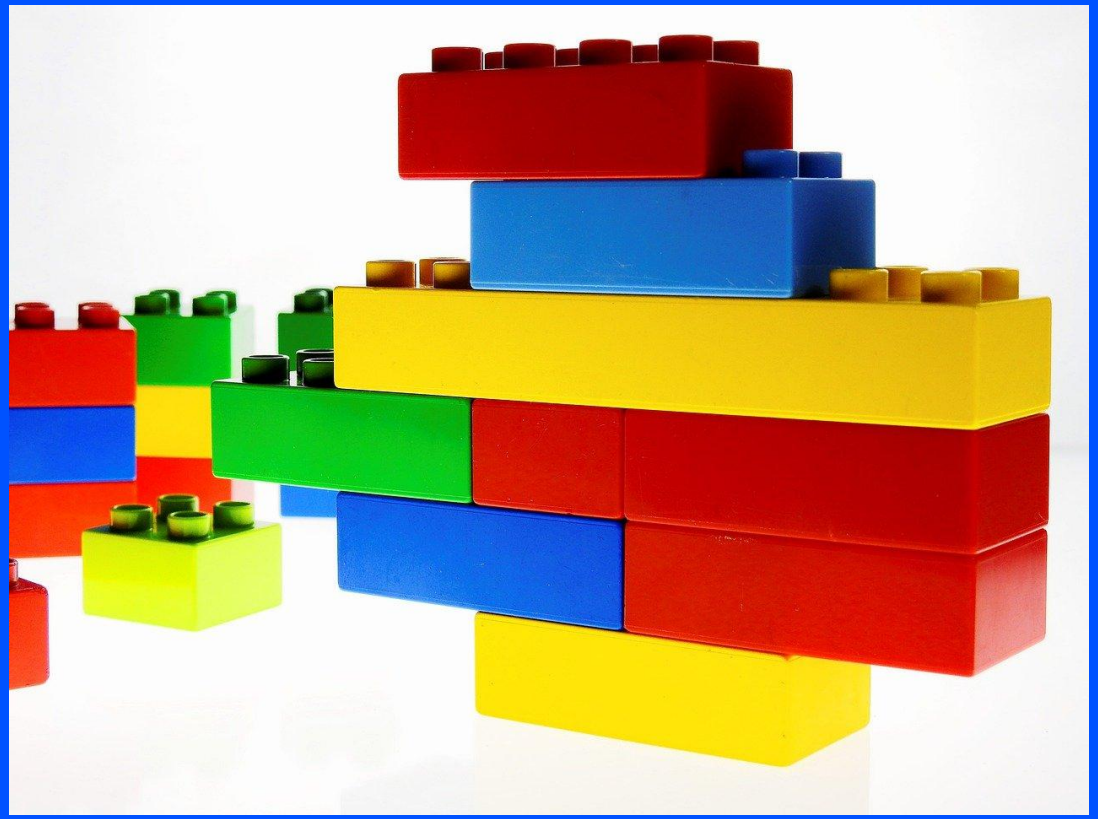
Blocks

PREFECT ASSOCIATE
CERTIFICATION

Configuration

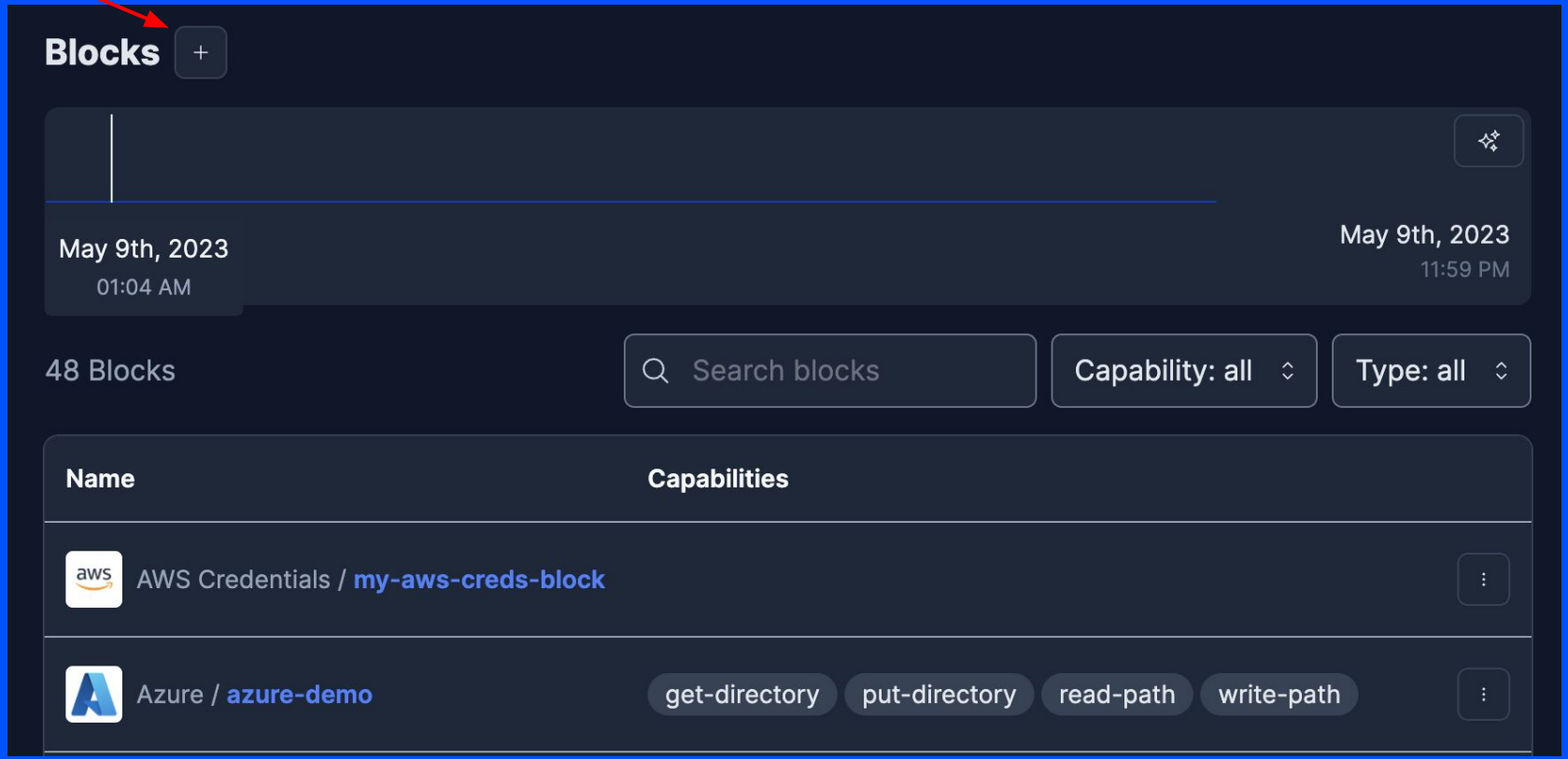
+

Code



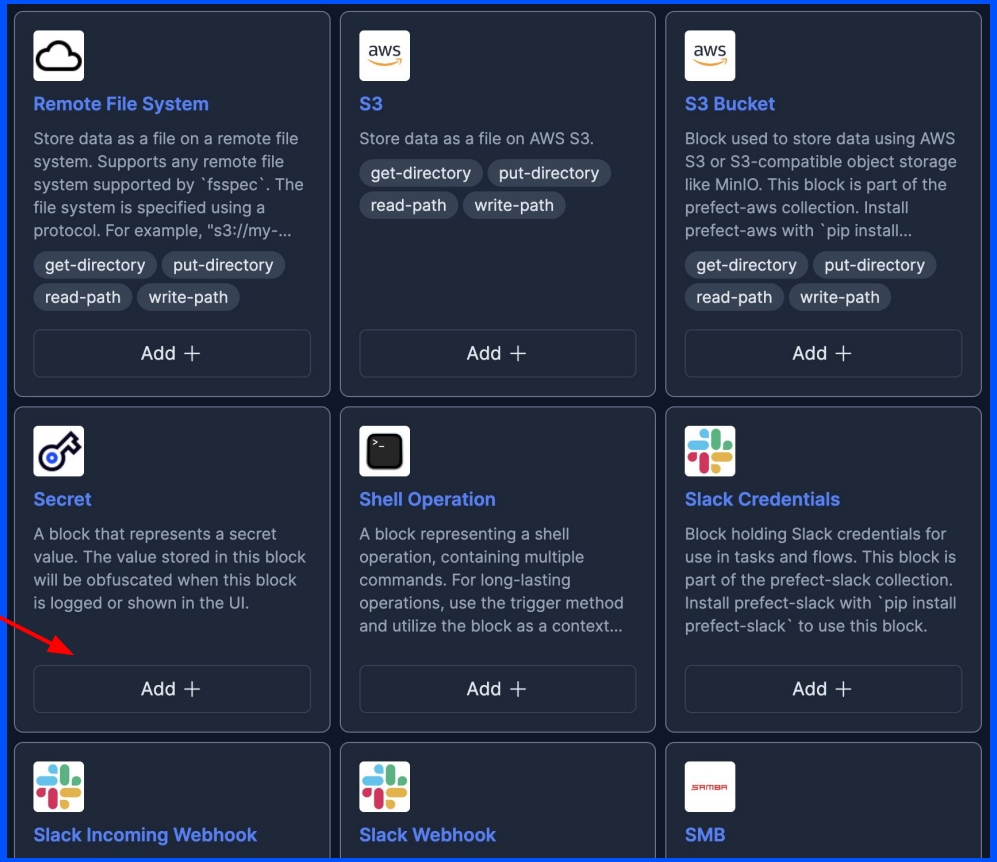
Blocks in UI

PREFECT ASSOCIATE
CERTIFICATION



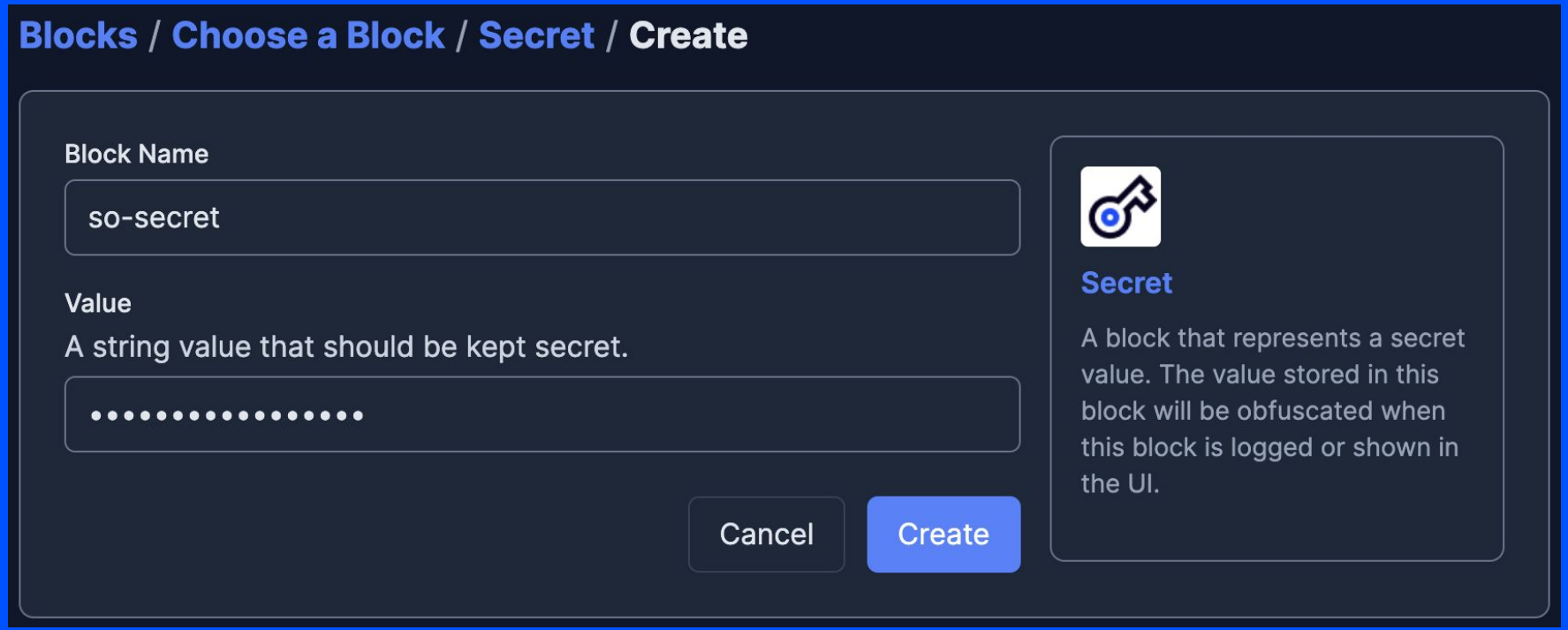
Select a Secret block

PREFECT ASSOCIATE
CERTIFICATION



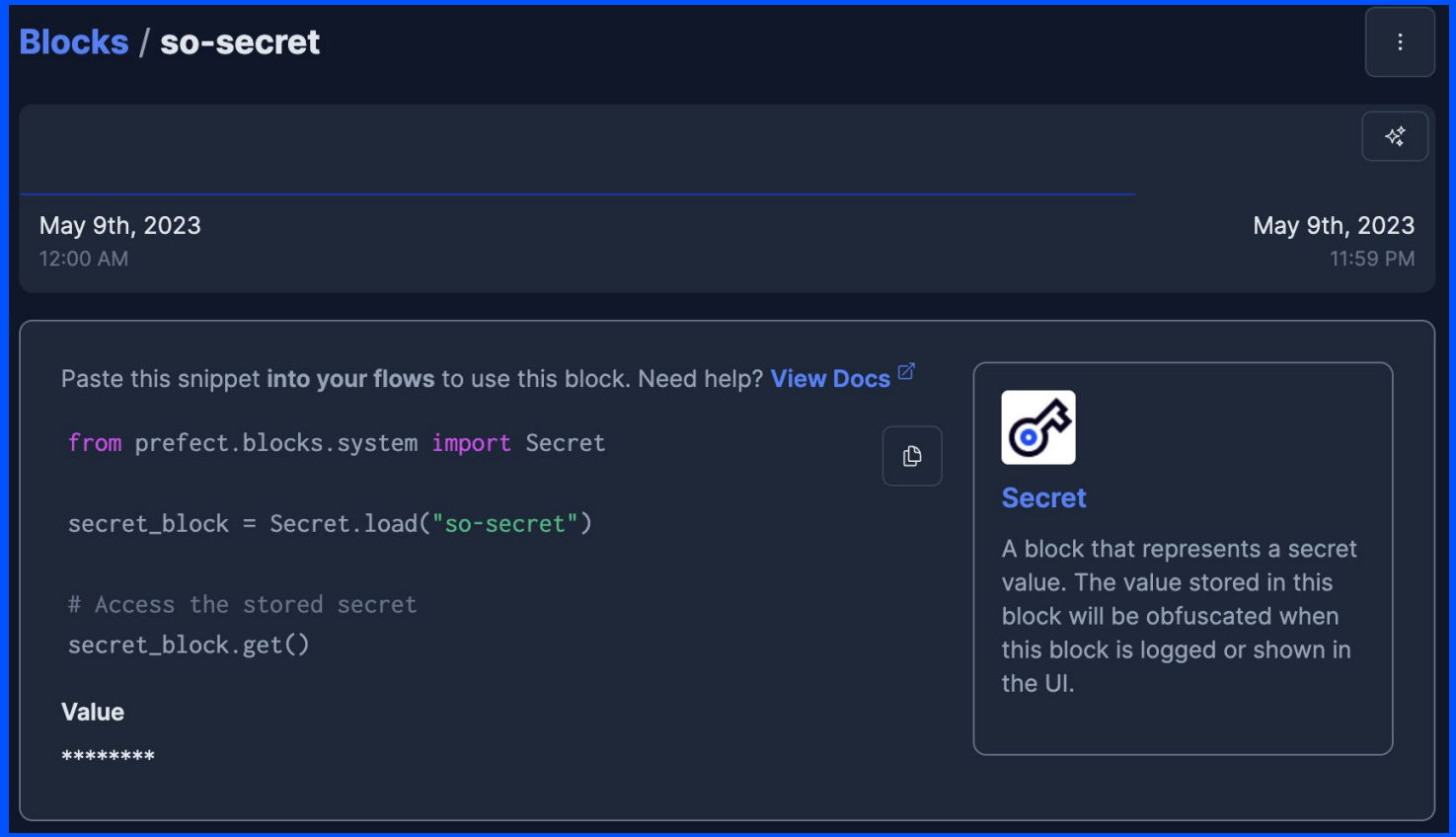
Create a Secret block

PREFECT ASSOCIATE CERTIFICATION



Use the Secret block

PREFECT ASSOCIATE
CERTIFICATION



Create a JSON block from code

PREFECT ASSOCIATE
CERTIFICATION

```
from prefect.blocks.system import JSON

autos = JSON(value=dict(cars=["tesla", "fiat", "chevy"], trucks=['rivian', 'ford']))
autos.save(name="json-block-ex", overwrite=True)
```

Blocks can be created from code or UI

Blocks

PREFECT ASSOCIATE
CERTIFICATION

Reusable, modular, configuration + code

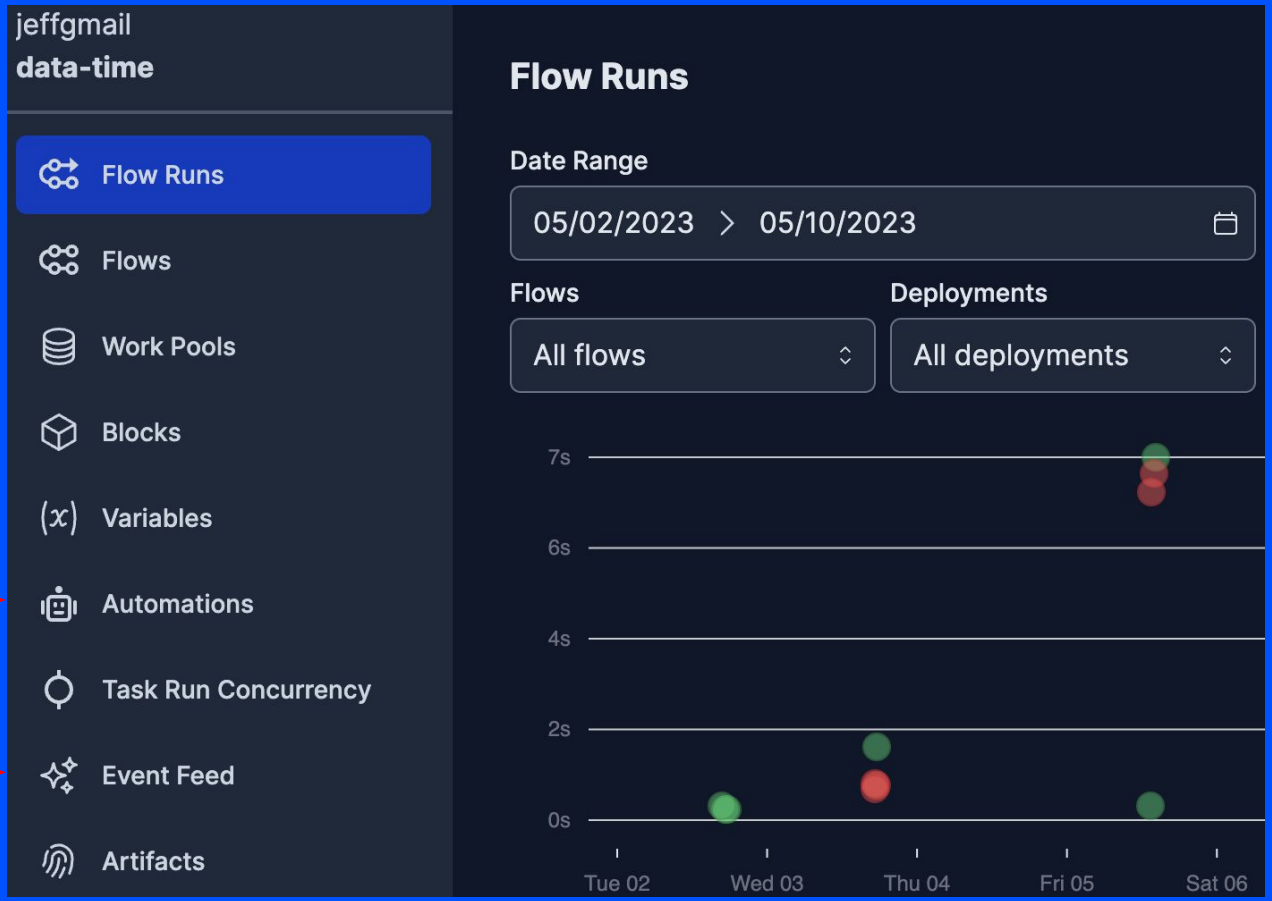
- Better than hard coding
- Better than environment variables
- Nestable
- Can create own types

PREFECT

Cloud Features

Automations

PREFECT ASSOCIATE
CERTIFICATION



PREFECT

Automations

Automations

Cloud only

Flexible framework

- If *Trigger* happens, do *Action*
- If *Trigger* doesn't happen in a time period, do *Action*

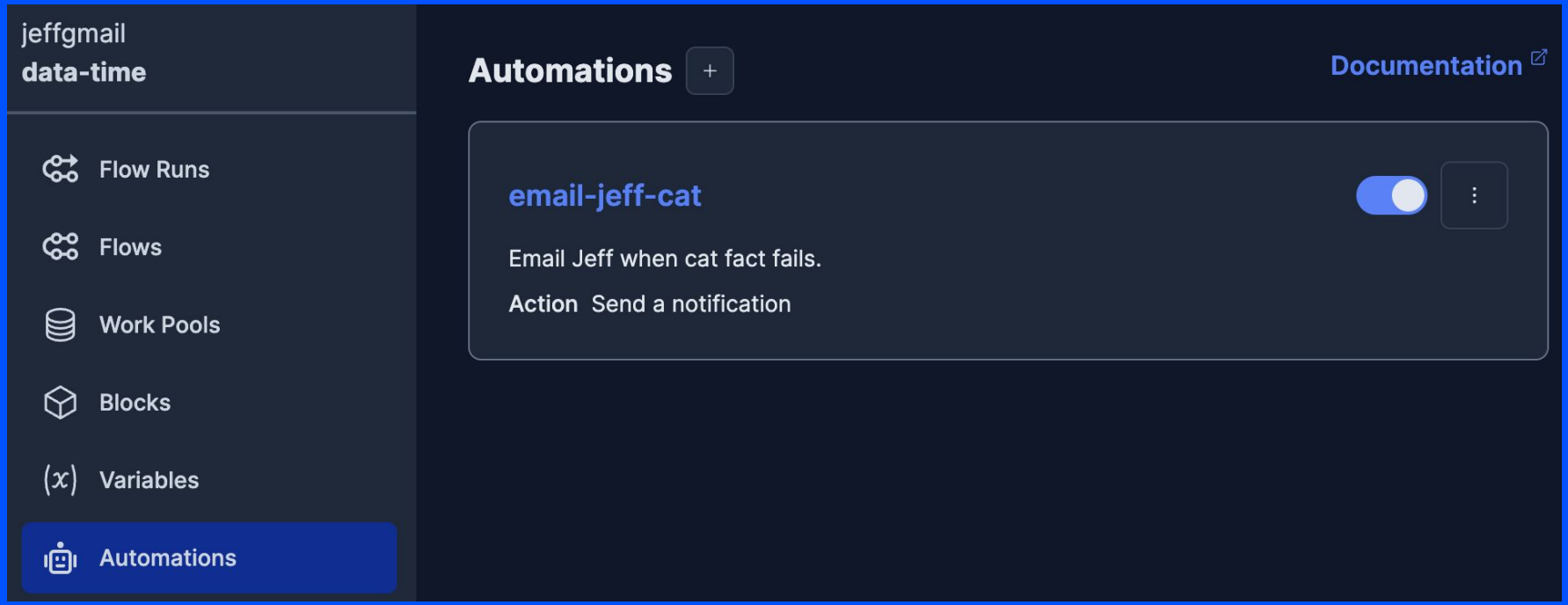
Notifications

PREFECT ASSOCIATE
CERTIFICATION

For example, send an email if a flow fails

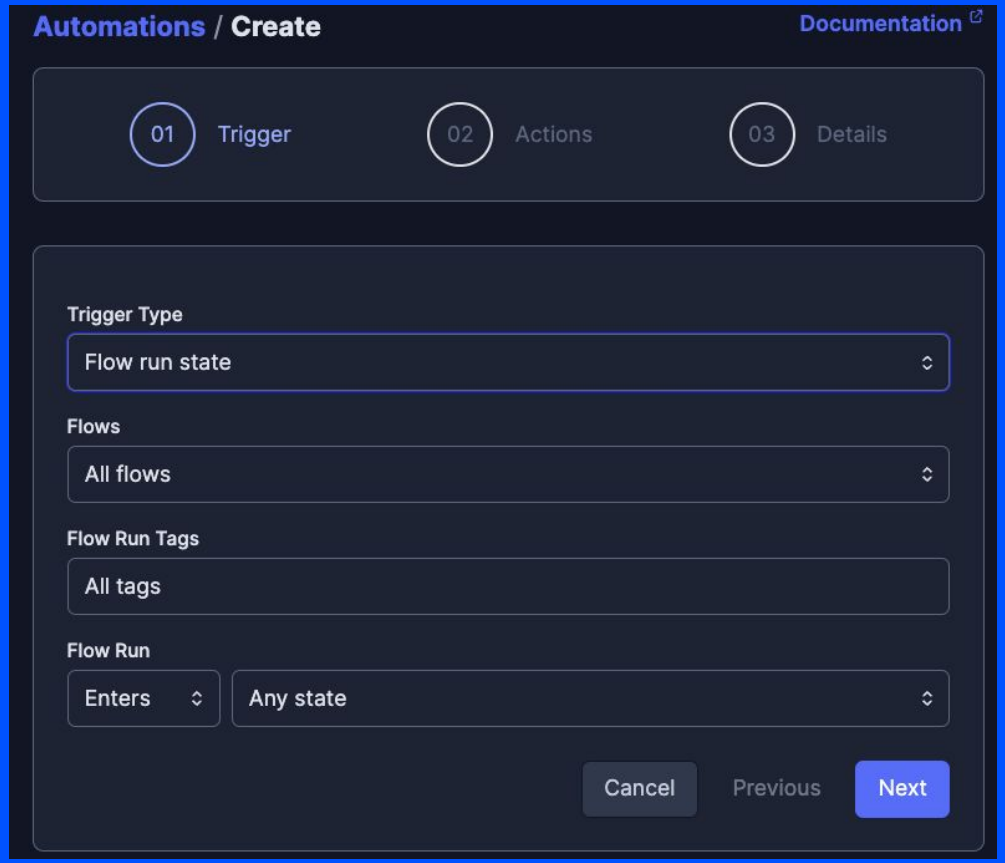
Automations

PREFECT ASSOCIATE
CERTIFICATION



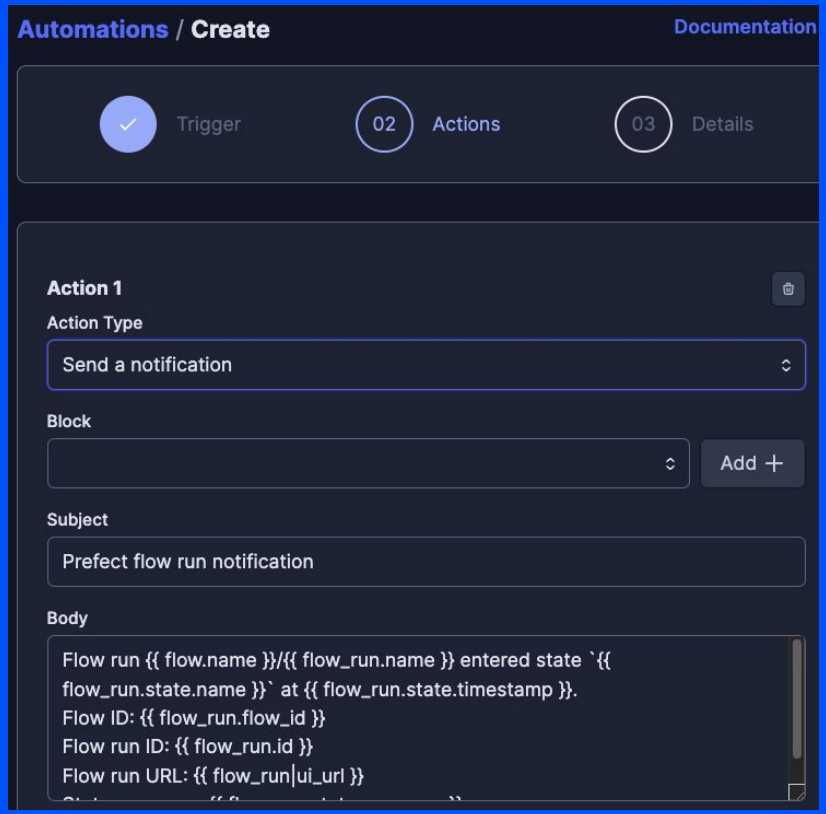
Automations

PREFECT ASSOCIATE
CERTIFICATION



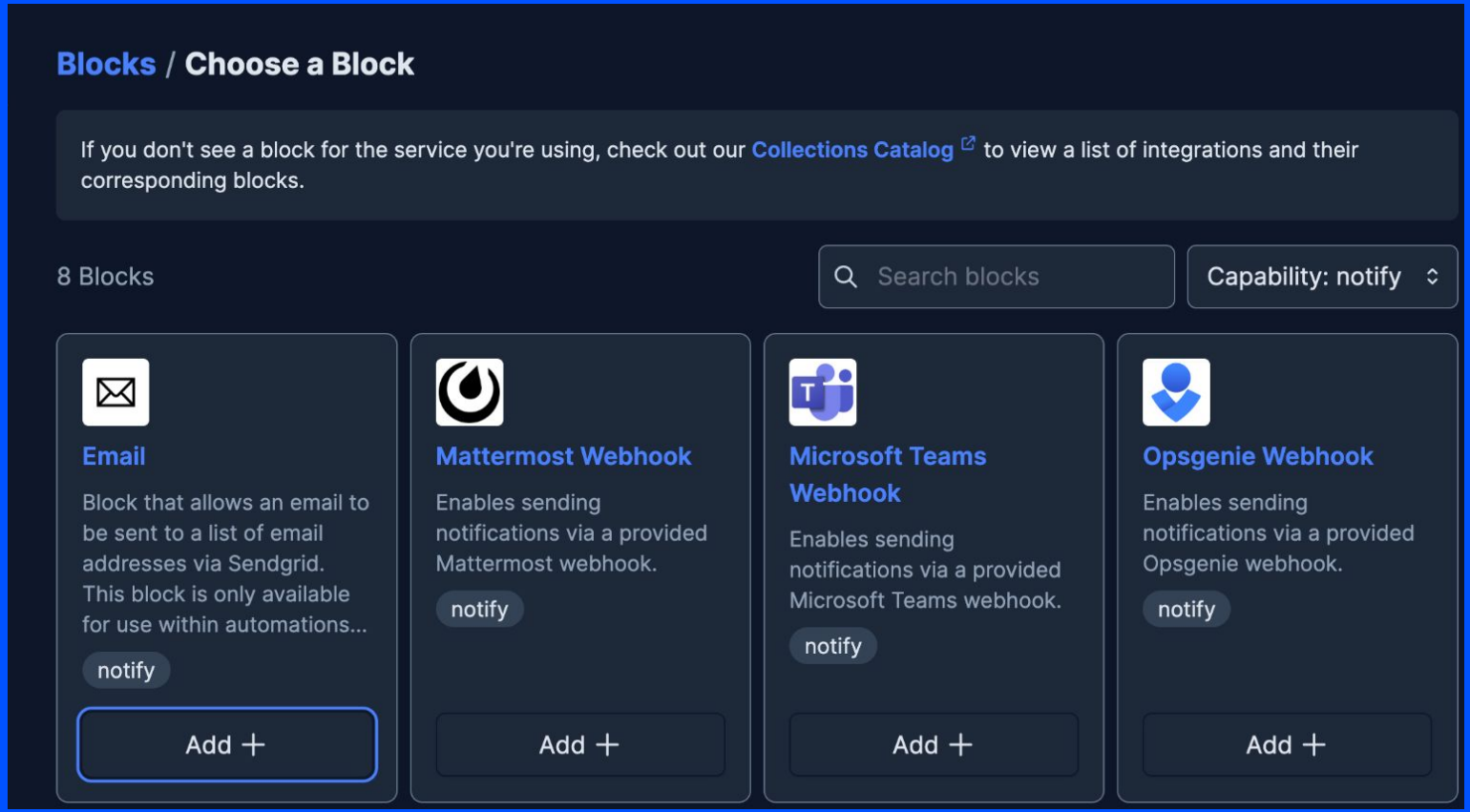
Automations

PREFECT ASSOCIATE
CERTIFICATION



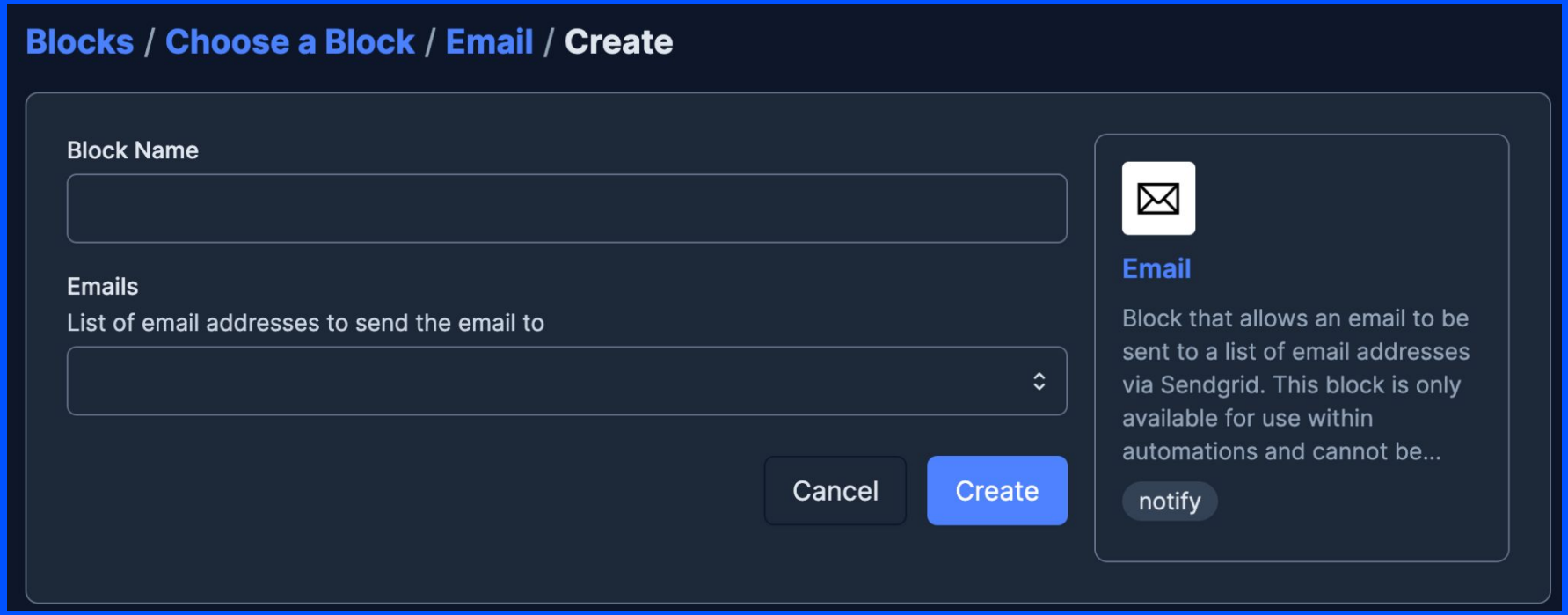
Automations

PREFECT ASSOCIATE CERTIFICATION



Automations

PREFECT ASSOCIATE
CERTIFICATION



PREFECT

Events

Events

PREFECT ASSOCIATE
CERTIFICATION

- A notification of a change
- A feed of activity recording what's happening


Prefect Technologies
prefect-prd-internal-tools
Flow Runs
Flows
Deployments
Work Pools
Blocks
Automations
Task Run Concurrency
Event Feed
Workspace Events
Beta
Block events are in Beta. To gather events from blocks, including storage and infrastructure events, enable client side events with prefect config set PREFECT_EXPERIMENTAL_ENABLE_EVENTS_CLIENT='True'
Related Resource
All resources
Events
All events
Mar 19th, 06:10 PM - Mar 22nd, 10:48 PM
Mar 17th, 2023
12:00 AM
1H
1D
1W
Mar 21st, 2023
06:32 PM
Mar 23rd, 2023
11:59 PM
Block Document collection-registry-github-token
5426 events
Block Document marvin-monday-slack-posts-k8s-job
1683 events
Block Document pd-oncall-slack-update-k8s-job
819 events
Block Document closed-won-k8s-job
756 events
Block Document collection-registry-update-job
460 events
Block Document collection-registry-result-storage
93 events
See More
10:30:52 PM
Mar 22nd, 2023
Flow run scheduled
prefect.flow-run.Scheduled
Resource
Flow run steady-ocelot
Related Resources
Flow Monday Flow
Deployment Marvin Monday Slack Posts
Work Queue internal-tools-cluster
1 other resource
10:30:28 PM
Mar 22nd, 2023
Block kubernetes job get client called
prefect.block.kubernetes-job.get_client.called
Resource
Block Document marvin-monday-slack-posts-k8s-job
Related Resources
1 resource
10:30:26 PM
Mar 22nd, 2023
Block kubernetes job get batch client called
prefect.block.kubernetes-job.get_batch_client.called
Resource
Block Document marvin-monday-slack-posts-k8s-job
Related Resources
1 resource
10:30:24 PM
Flow run completed

Events

Can represent

- API calls
- state transitions
- changes in environment

Event Feeds

PREFECT ASSOCIATE
CERTIFICATION

Power several Cloud features

- flow run logs
- audit logs
- automations

103 Recap

PREFECT ASSOCIATE
CERTIFICATION

You've learned

- about blocks, including Secret, JSON & Notification blocks
- how to use Prefect Cloud
- Prefect Cloud automations and events

PREFECT

Lab 103

103 Lab

PREFECT ASSOCIATE
CERTIFICATION

- Use a previously made flow
- Use Prefect Cloud
- Make an email notification automation for flow run completion
- Run the flow a few times
- Check out the event feed in the UI

PREFECT

MODULE

104 - Deployments & Projects

104 Agenda

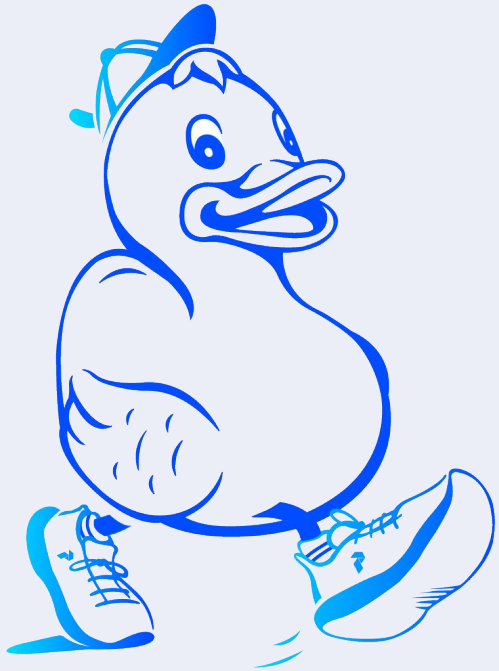
PREFECT ASSOCIATE
CERTIFICATION

- Deployments
- Projects
- Create a deployment from a project
- Create a work pool and start a worker
- Run a deployment

Deployments

Minerva wants to step up her Prefect orchestration game.

Deployments bring more features and are needed for production.



Deployments

PREFECT ASSOCIATE
CERTIFICATION

A deployment contains info about all the things your flow needs to run in production.

Deployments: ETL code

PREFECT ASSOCIATE
CERTIFICATION

```python
@task
def fetch_cat_fact():
    return httpx.get("https://catfact.ninja/fact?max_length=140").json()["fact"]

@task
def formatting(fact: str):
    return fact.title()

@task
def write_fact(fact: str):
    with open("fact.txt", "w+") as f:
        f.write(fact)
    return "Success!"
```

Deployments: ETL code

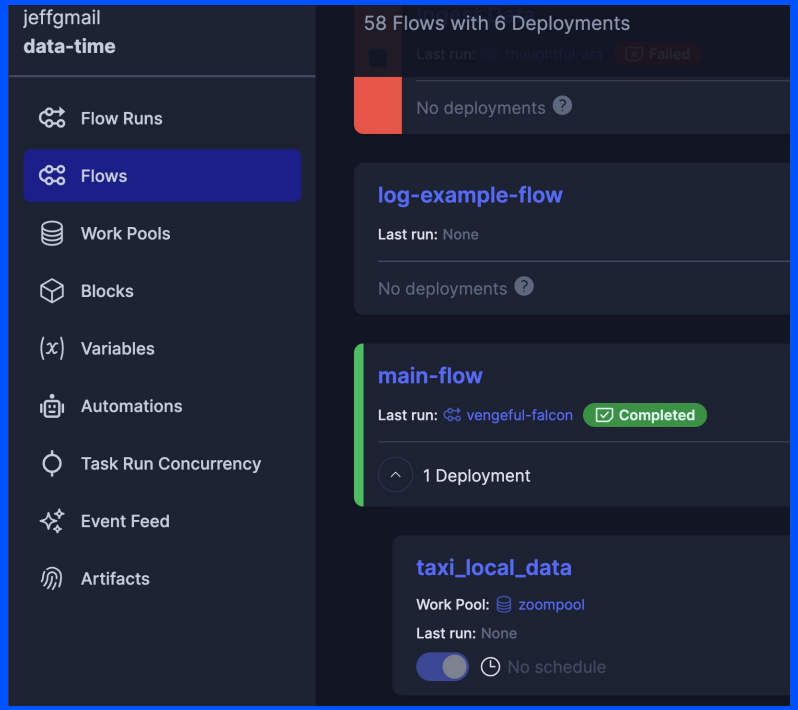
PREFECT ASSOCIATE
CERTIFICATION

```python
@flow
def pipe():
    fact = fetch_cat_fact()
    formatted_fact = formatting(fact)
    msg = write_fact(formatted_fact)
    print(msg)
```

Deployments in the UI

PREFECT ASSOCIATE CERTIFICATION

The deployment will live on the server



PREFECT

Prefect Projects

Prefect project

PREFECT ASSOCIATE
CERTIFICATION

Directory with the following files in the root

1. *prefect.yaml*
2. *deployment.yaml*
3. *.prefect/*
4. *.prefectignore*

Files describe how to prepare one or more flow deployments

Prefect Project

PREFECT ASSOCIATE
CERTIFICATION

prefect.yaml

- steps to prepare a deployment from the project
- instructions to create the execution environment for a deployment run

```yaml
# generic metadata
prefect-version: null
name: null

# preparation steps
build: null
push: null

# runtime steps
pull: null
```

Prefect Project - deployment.yaml

PREFECT ASSOCIATE
CERTIFICATION

Describes base settings for a deployment produced from the project.

```yaml
deployments:
- name: null
  version: null
  tags: []
  description: null
  schedule: {}
  flow_name: null
  entrypoint: null
  parameters: {}
  work_pool:
    name: null
    work_queue_name: null
    job_variables: {}
```

Prefect Project

PREFECT ASSOCIATE
CERTIFICATION

.prefect/

- hidden directory where Prefect will store workflow metadata

Prefect Project

PREFECT ASSOCIATE
CERTIFICATION

.prefectignore

- like *.gitignore*
- use it to keep files out of a deployment

Prefect Project

PREFECT ASSOCIATE
CERTIFICATION

Create a project with

prefect project init

in the directory you want to be the root of a project

Prefect Project - prefect.yaml

PREFECT ASSOCIATE
CERTIFICATION

```yaml
# Generic metadata about this project
name: 2023-05-15_ppcc_demos
prefect-version: 2.10.8

# build section allows you to manage and build docker images
build: null

# push section allows you to manage if and how this project is uploaded to remote locations
push: null

# pull section allows you to provide instructions for cloning this project in remote locations
pull:
- prefect.projects.steps.git_clone_project:
    repository: https://github.com/discdiver/ppcc-2023-05.git
    branch: main
    access_token: null
```

Prefect Project *prefect project init*

PREFECT ASSOCIATE
CERTIFICATION

Uses environment attributes to populate *prefect.yaml*

- *name*: from directory (*2023-05-15_ppcc_demos*)
- *prefect-version*: from active environment (*2.10.8*)
- *pull* step (repository & branch): from git repo

PREFECT

Workers

Workers

- Wait for scheduled flow runs from work pools on the server
- Must match
- Client side

```
prefect worker start -p my_pool -t process
```

PREFECT

Work Pools

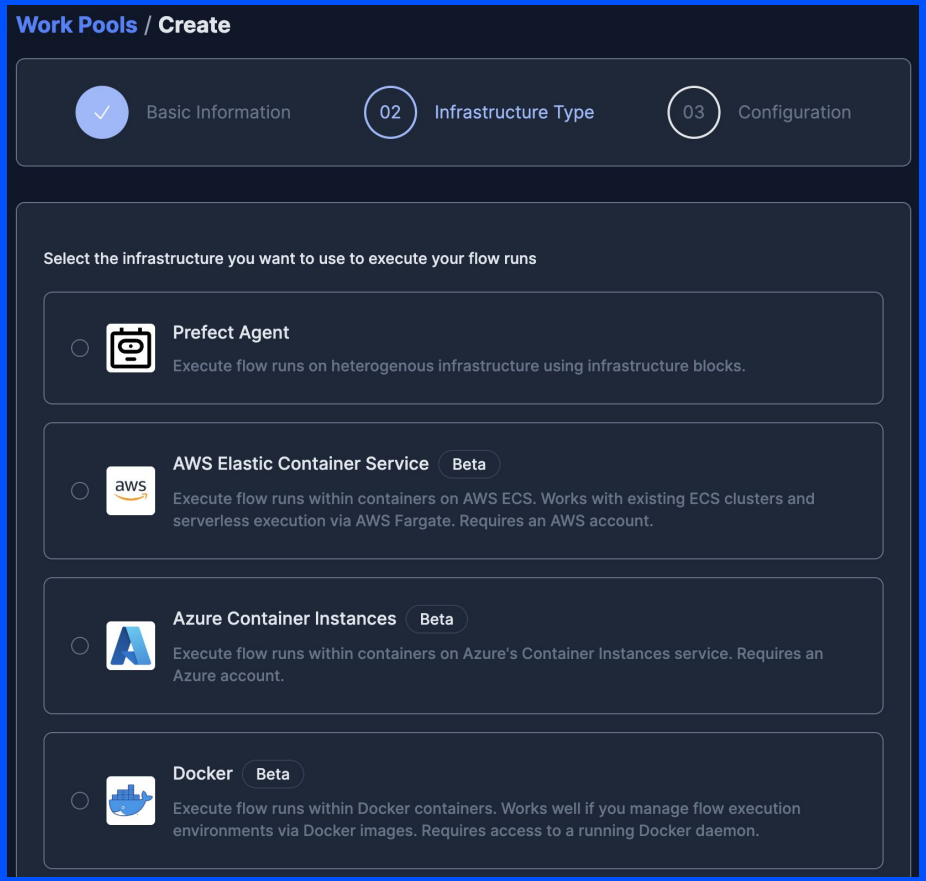
Work Pools

PREFECT ASSOCIATE
CERTIFICATION

- Where scheduled flow runs go
- Typed by infrastructure (e.g. Docker, Kubernetes)
- Can be created via UI or CLI
- Workers poll work pools for work

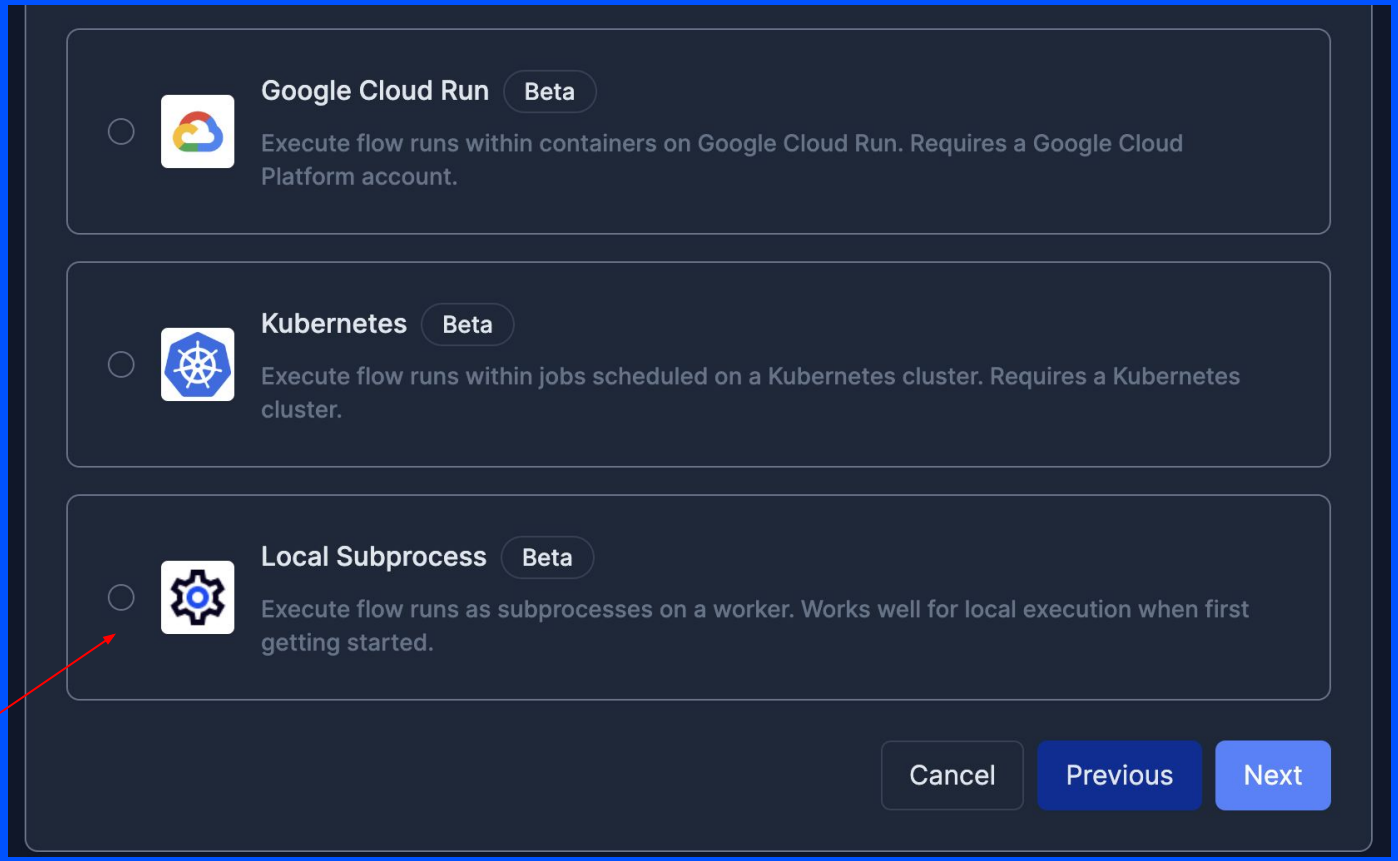
Work Pools

PREFECT ASSOCIATE
CERTIFICATION



Work Pools

PREFECT ASSOCIATE
CERTIFICATION



PREFECT

Project Deploy

Project Deploy

PREFECT ASSOCIATE
CERTIFICATION

Send deployment to server

```
prefect deploy 104/flows.py:pipe
```

Project Deploy

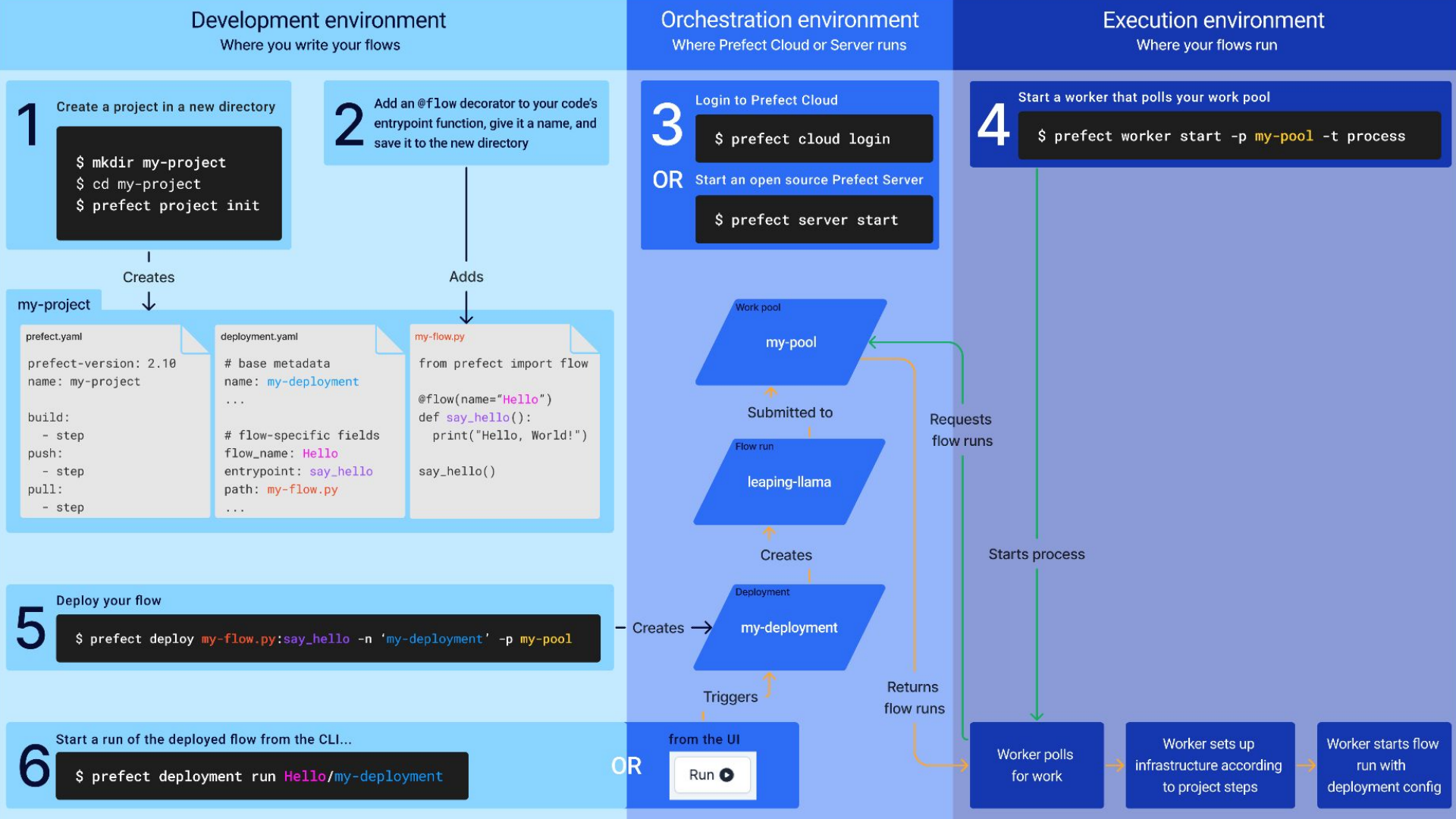
PREFECT ASSOCIATE CERTIFICATION

Interactive prompts:

- create deployment name
- choose work pool

```
? Deployment name: interactive
? Which work pool would you like to deploy this flow to? [Use arrows to move; enter to select]
```

	Work Pool Name	Infrastructure Type	Description
	docker-work	docker	
	local-work	process	
>	**my-pool**	process	
	prod-pool	kubernetes	
	staging-pool	kubernetes	
	zoompool	process	


Development environment
Where you write your flows
Orchestration environment
Where Prefect Cloud or Server runs
Execution environment
Where your flows run
1
Create a project in a new directory
$ mkdir my-project
$ cd my-project
$ prefect project init
2
Add an @flow decorator to your code's entrypoint function, give it a name, and save it to the new directory
3
Login to Prefect Cloud
$ prefect cloud login
OR
Start an open source Prefect Server
$ prefect server start
4
Start a worker that polls your work pool
$ prefect worker start -p my-pool -t process
Creates
Adds
my-project
prefect.yaml
prefect-version: 2.10
name: my-project
build:
- step
push:
- step
pull:
- step
deployment.yaml
base metadata
name: my-deployment
...
flow-specific fields
flow_name: Hello
entrypoint: say_hello
path: my-flow.py
...
my-flow.py
from prefect import flow
@flow(name="Hello")
def say_hello():
print("Hello, World!")
say_hello()
Work pool
my-pool
Submitted to
Flow run
leaping-llama
Creates
Deployment
my-deployment
Triggers
Requests flow runs
Returns flow runs
Starts process
5
Deploy your flow
$ prefect deploy my-flow.py:say_hello -n 'my-deployment' -p my-pool
Creates
6
Start a run of the deployed flow from the CLI...
$ prefect deployment run Hello/my-deployment
OR
from the UI
Run
Worker polls for work
Worker sets up infrastructure according to project steps
Worker starts flow run with deployment config

What just happened?

PREFECT ASSOCIATE
CERTIFICATION

- Deployment *my-project-deploy* created on Prefect Cloud.
- Work pool *my_pool* created on Prefect Cloud.
- Worker started on local machine.
- Worker process polls Prefect Cloud, looking for scheduled work in the *my_pool* work pool.

Schedule a run

- Running worker finds scheduled work in *my_pool* work pool.
- Worker and work pool are typed. *process* in this case.
- Worker creates a local subprocess to kick off flow run.
- Flow code cloned from GitHub into temporary directory.
- Temporary directory deleted.

Run a deployment from the UI

PREFECT ASSOCIATE CERTIFICATION

From the *Flows* menu, expand the deployment and choose *Quick run* from the three dot menu

View the flow run logs in the UI

PREFECT ASSOCIATE CERTIFICATION

Feb 14th, 2023

INFO	Downloading flow code from storage at '/Users/jeffhale/Desktop/prefect/demos/pacc/pacc-nyc-demos-2023-02-13'	04:31:29 PM
INFO	Created task run 'fetch_cat_fact-0' for task 'fetch_cat_fact'	04:31:31 PM
INFO	Executing 'fetch_cat_fact-0' immediately...	04:31:31 PM
INFO	Finished in state Completed()	04:31:32 PM fetch_cat_fact-0
INFO	Created task run 'formatting-0' for task 'formatting'	04:31:32 PM
INFO	Executing 'formatting-0' immediately...	04:31:32 PM
INFO	Finished in state Completed()	04:31:33 PM formatting-0
INFO	Created task run 'write_fact-0' for task 'write_fact'	04:31:33 PM
INFO	Executing 'write_fact-0' immediately...	04:31:33 PM
INFO	Finished in state Completed()	04:31:34 PM write_fact-0

104 Recap

PREFECT ASSOCIATE
CERTIFICATION

You've learned how to build and apply *Deployments* with Prefect Projects.

You've learned how to start an *Worker* that polls a *Work Pool* looking for scheduled *Flow Runs*.

You've seen how to inspect a *Flow Run* from the UI.

PREFECT

Lab 104

104 Lab

PREFECT ASSOCIATE
CERTIFICATION

Use your flow from 103 Lab

- If you have flow function parameters, create default arguments in your flow function
- Make a project
- Start a worker
- Run a deployment from the UI
- Check it out in the UI

PREFECT

MODULE

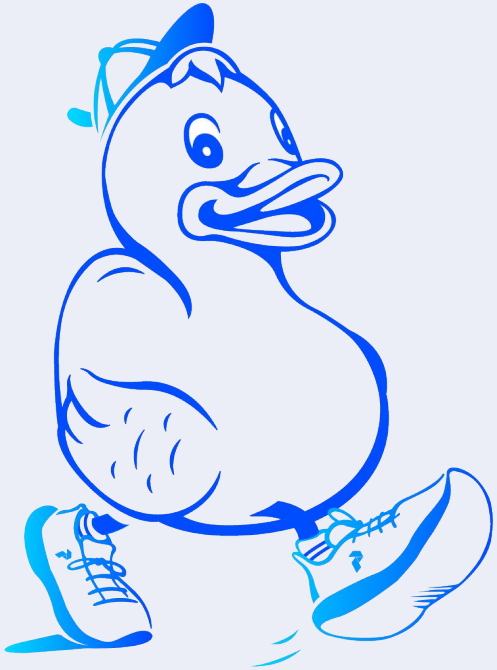
105 - Deployments & Artifacts

105 Agenda

- Parameters
- Scheduling ⏳
- Pausing and resuming schedules ⏯️
- Artifacts

Parameters need values

If your flow function has params and no defaults, you must feed it (give it values).



PREFECT

Parameters

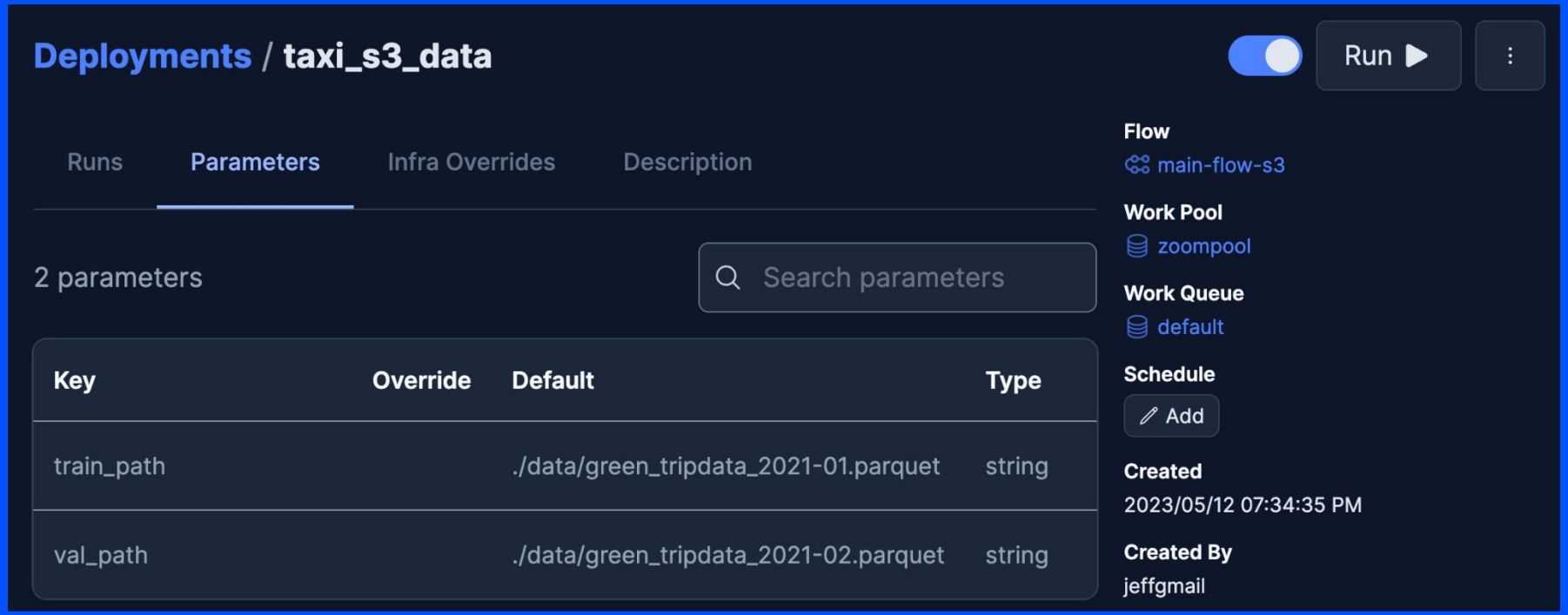
Parameters - argument values for flow function

Options

- make default arguments in flow function definition
- adjust in deployment.yaml
- customize in UI when run flow
- customize in CLI when run flow

Parameters in the UI

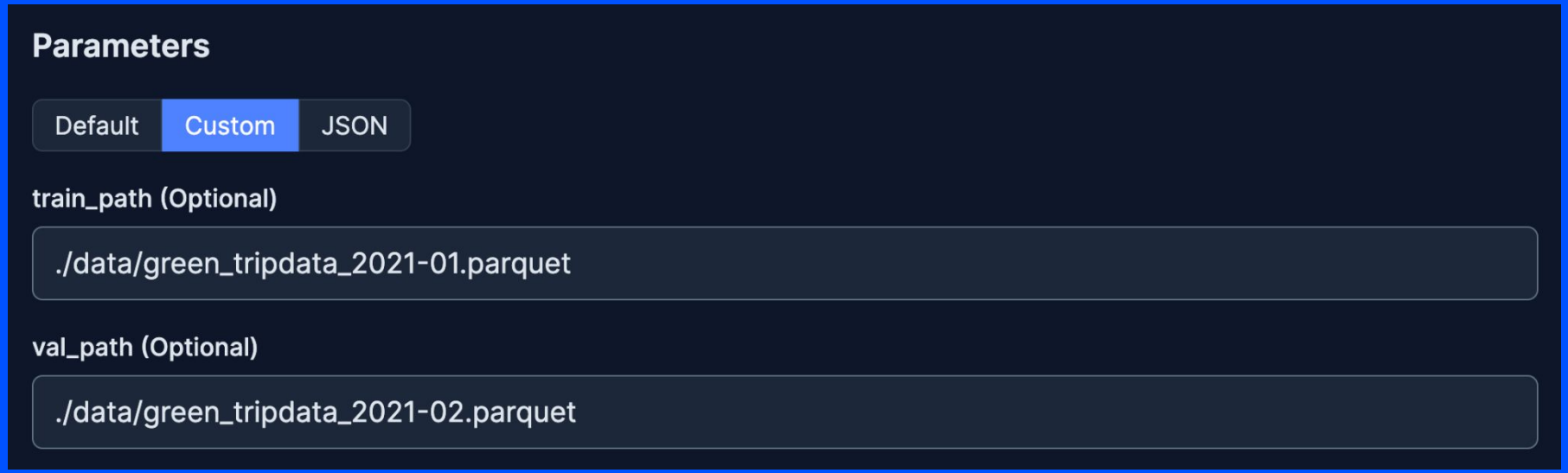
PREFECT ASSOCIATE CERTIFICATION



Parameters

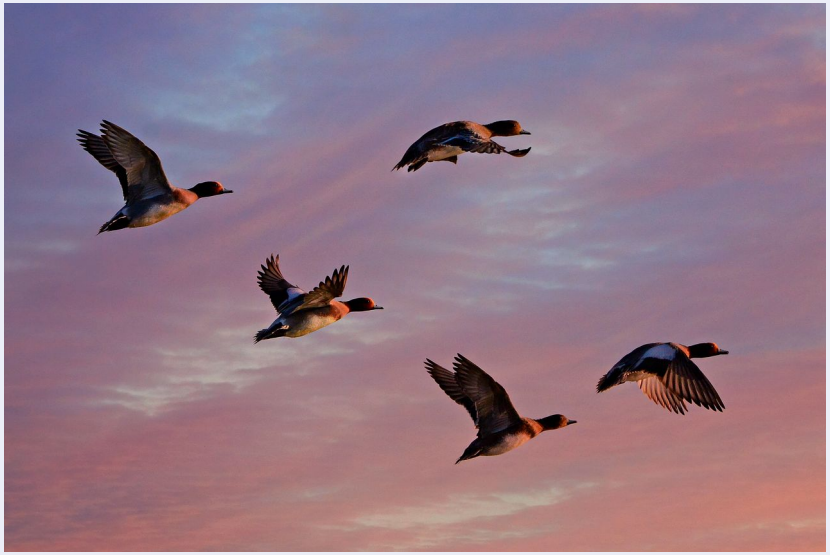
PREFECT ASSOCIATE
CERTIFICATION

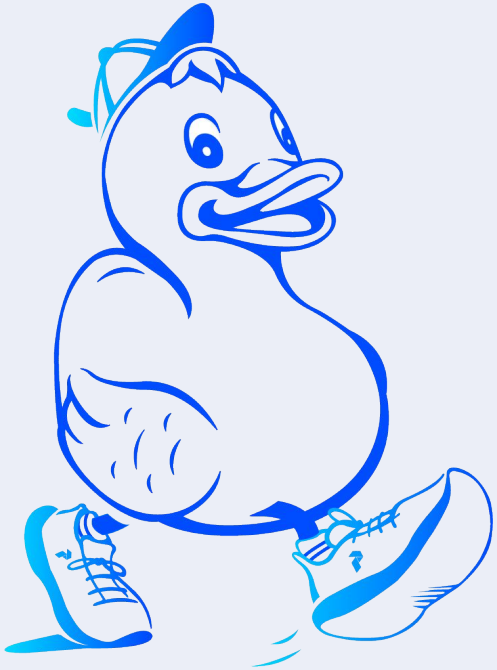
Collaborators can run with custom values



Scheduling

Like other ducks, Minerva likes things to happen on time.





PREFECT

Scheduling

Scheduling

1. When you *prefect deploy*, specified in the *deployment.yaml* file
2. When you *prefect deploy*, in the cli command
3. After deployment creation in the UI
4. After deployment creation in the CLI

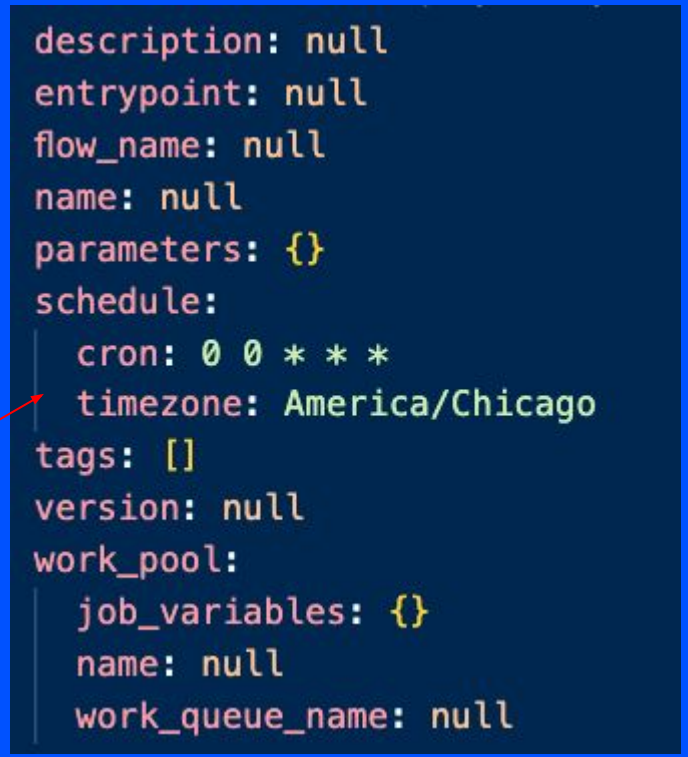
Schedule types

PREFECT ASSOCIATE
CERTIFICATION

- Interval
- Cron
- RRule

Add a schedule in *deployment.yaml*

Schedule created in Cloud when you run *prefect deploy*



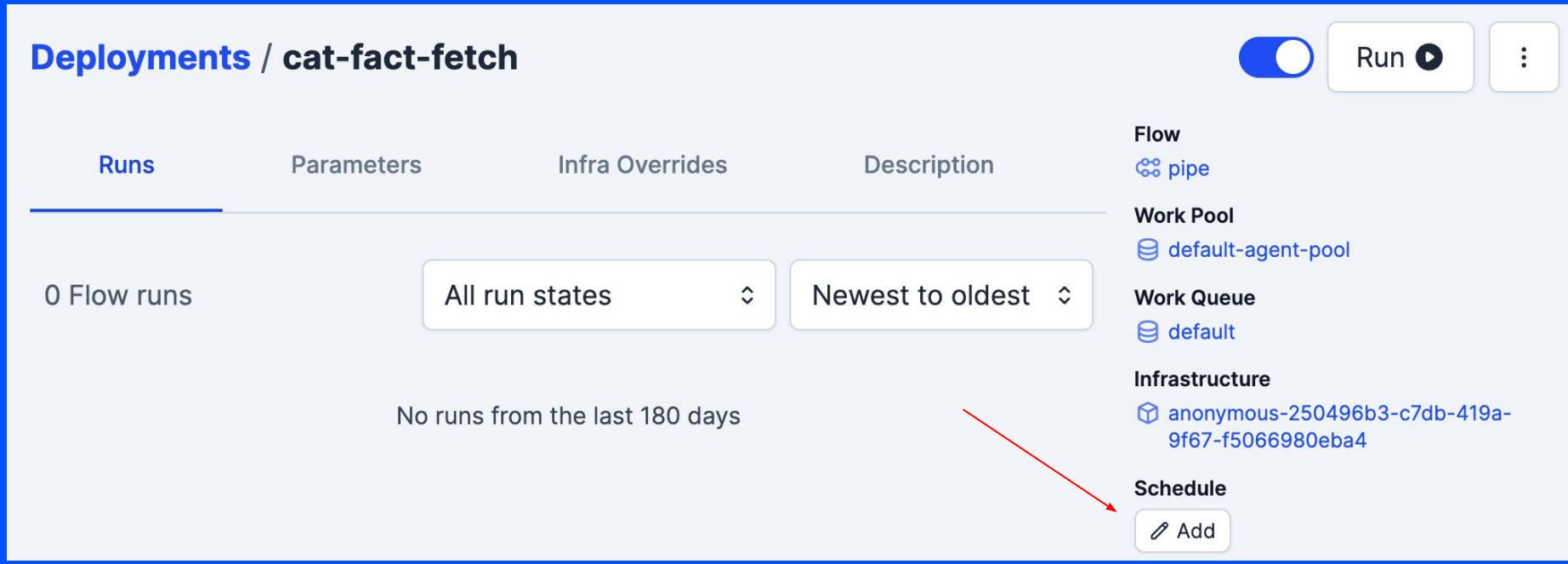
Add a schedule in the *prefect deploy* command

PREFECT ASSOCIATE
CERTIFICATION

```
prefect deploy -n my_deploy -p
my_work_pool --interval 50
```

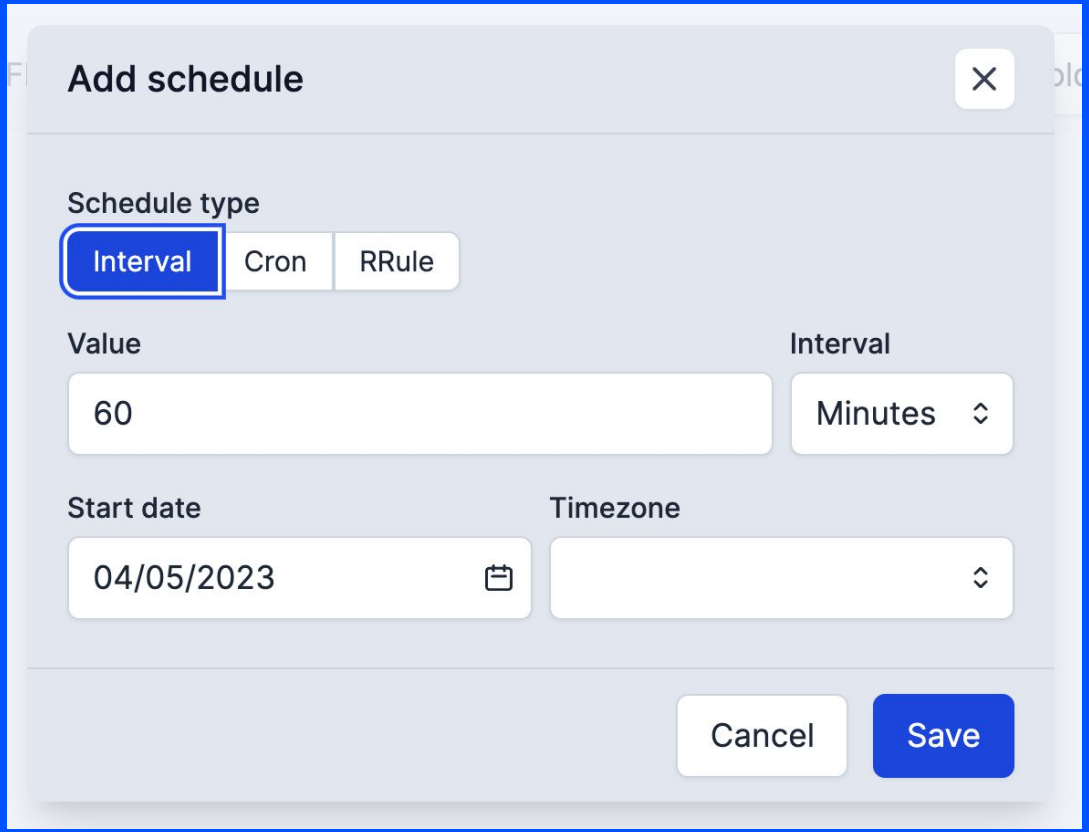
Scheduling from the UI

PREFECT ASSOCIATE CERTIFICATION



Scheduling from the UI

PREFECT ASSOCIATE
CERTIFICATION



Scheduling from CLI

PREFECT ASSOCIATE
CERTIFICATION

```
prefect deployment set-schedule
weatherflow.py:fetch_weather --interval 40
```

Scheduling RRule

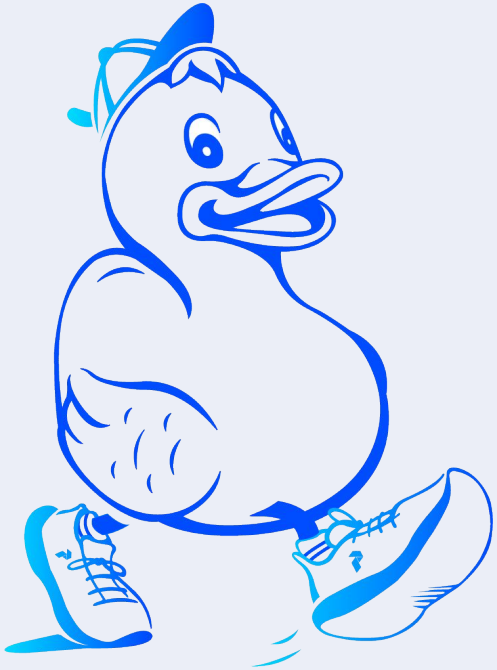
PREFECT ASSOCIATE CERTIFICATION

RRule cheat sheet: jakubroztocil.github.io/rrule/

Or ask Marvin (another Prefect package) *pip install marvin*

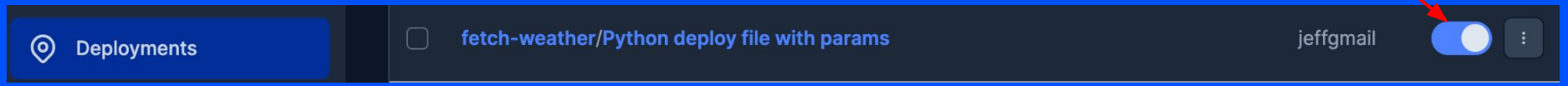
```python
from marvin import ai_fn

@ai_fn
def rrule(text: str) -> str:
    """
    Generate valid RRULE strings from a natural language description of an event
    """
    yield pendulum.now.isoformat()

rrule('every hour from 9-6 on thursdays')
# "RRULE:FREQ=WEEKLY;BYDAY=TH;BYHOUR=9,10,11,12,13,14,15,16;BYMINUTE=0;BYSECOND=0"
```

Pausing and restarting schedules



Pause schedule from UI

PREFECT ASSOCIATE
CERTIFICATION



Pause/Resume schedule from CLI

Pause ⏸️

```
prefect deployment pause-schedule
pipe_flow/"RRule Scheduled Deployment"
```

Resume ▶️

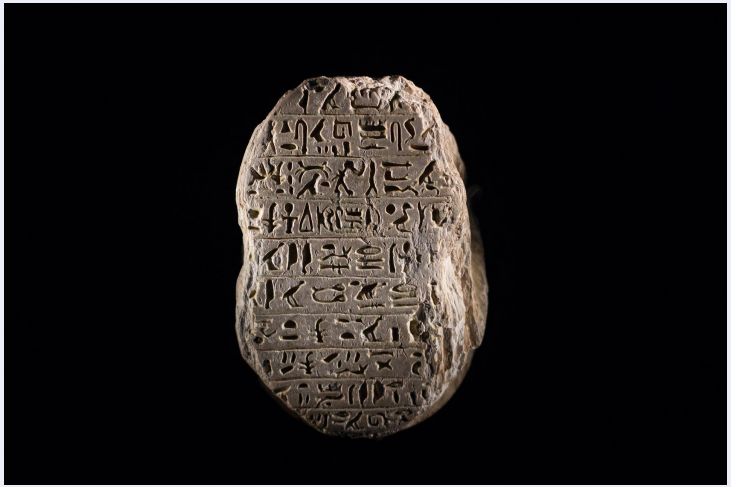
```
prefect deployment resume-schedule
pipe_flow/"RRule Scheduled Deployment"
```

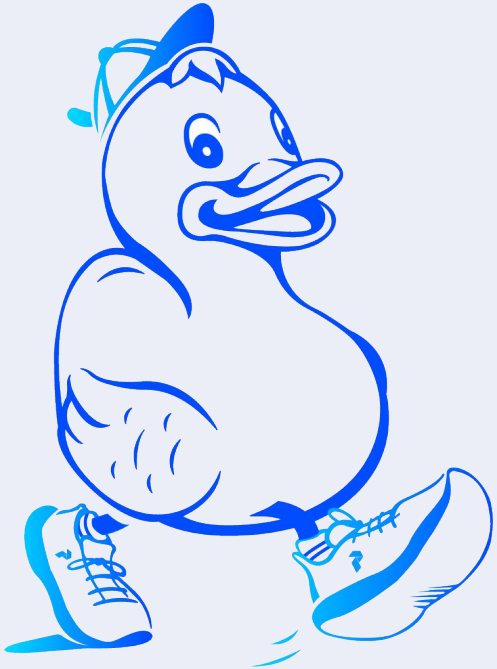
PREFECT

Artifacts

Artifacts

Persisted outputs such as Markdown, tables, or links.





Artifacts

PREFECT ASSOCIATE
CERTIFICATION

- Prettier output
- Meant for human consumption
  - Data quality checks
  - Documentation
- Not the main data transform result of a flow

Artifacts - Markdown example

PREFECT ASSOCIATE
CERTIFICATION

```python
import httpx
from prefect import flow, task
from prefect.artifacts import create_markdown_artifact

@task
def mark_it_down(temp):
    markdown_report = f"""# Weather Report

## Recent weather

| Time        | Revenue |
|:------------|-------:|
| Now | {temp} |
| In 1 hour       | {temp + 2} |
"""
    create_markdown_artifact(
        key="weather-report",
        markdown=markdown_report,
        description="Very scientific weather report",
    )
```

Artifacts - Markdown example

PREFECT ASSOCIATE
CERTIFICATION

Access from Flow Runs or *Artifacts* page

105 Recap

PREFECT ASSOCIATE
CERTIFICATION

You've seen how to work with deployments including how to:

- set parameters in several locations
- create schedules from the UI & CLI
- pause and resume schedules ⏯️
- create artifacts

PREFECT

Lab 105

105 Lab

PREFECT ASSOCIATE
CERTIFICATION

- Create a deployment with an entrypoint flow with several parameters
- Edit the deployment from the UI to override the parameters
- Add two types of schedules - add each a different way (CLI, UI)
- Pause the schedules
- Create two kinds of artifacts in your flow code code
- See your artifacts in the UI

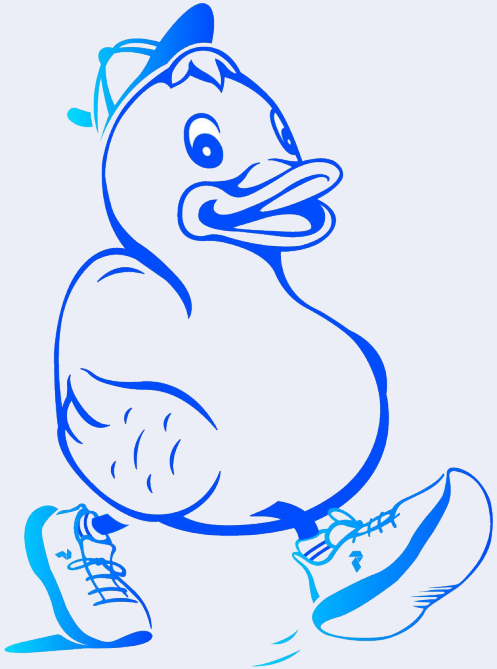
PREFECT

MODULE

201 - Integrations & Docker

Integrations

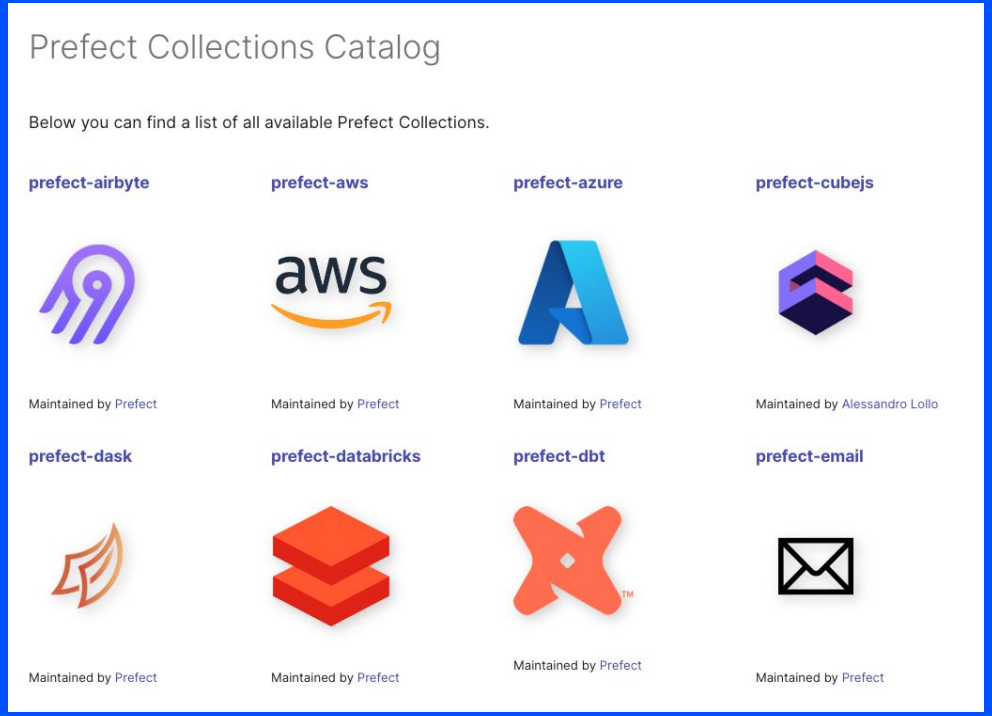
Ducks integrate well.



Integrations

PREFECT ASSOCIATE
CERTIFICATION

docs.prefect.io/integrations/catalog/



Integrations

PREFECT ASSOCIATE
CERTIFICATION

Python packages that add convenience

- 40+ integrations
- Template to create your own
- Can contribute to the community

Integrations

Prefect is Python-based and designed for flexibility

Use with most any Python library - no special integration required 🎉

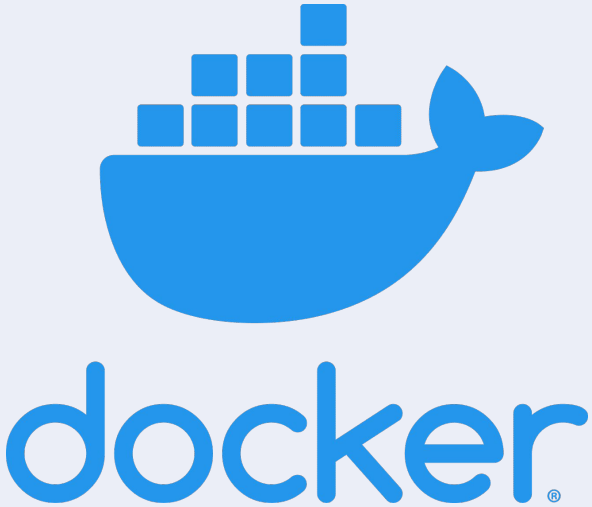
201 Agenda

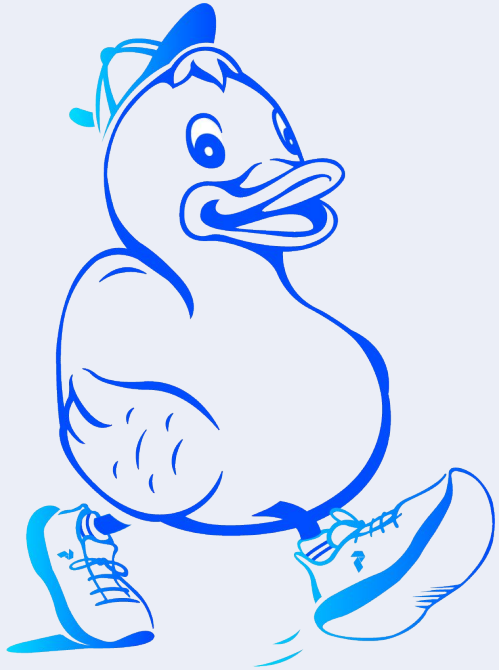
PREFECT ASSOCIATE
CERTIFICATION

Projects, Workers & Work Pools with Docker

Docker

Little known fact: ducks are all about containerization with Docker.





Why use Docker?

- Same operating environment everywhere
- Lighter weight than a VM
- Linux (generally)
- Portable
- Very popular

Docker

PREFECT ASSOCIATE
CERTIFICATION

- Prefect provides base Docker images
- Can customize base image
- Read about choosing images at docs.prefect.io/latest/concepts/infrastructure/#standard-python

Docker

PREFECT ASSOCIATE
CERTIFICATION

Prerequisites

- Docker *installed* & running
- *prefect-docker* package installed

PREFECT

Work pools - Docker

Flow run

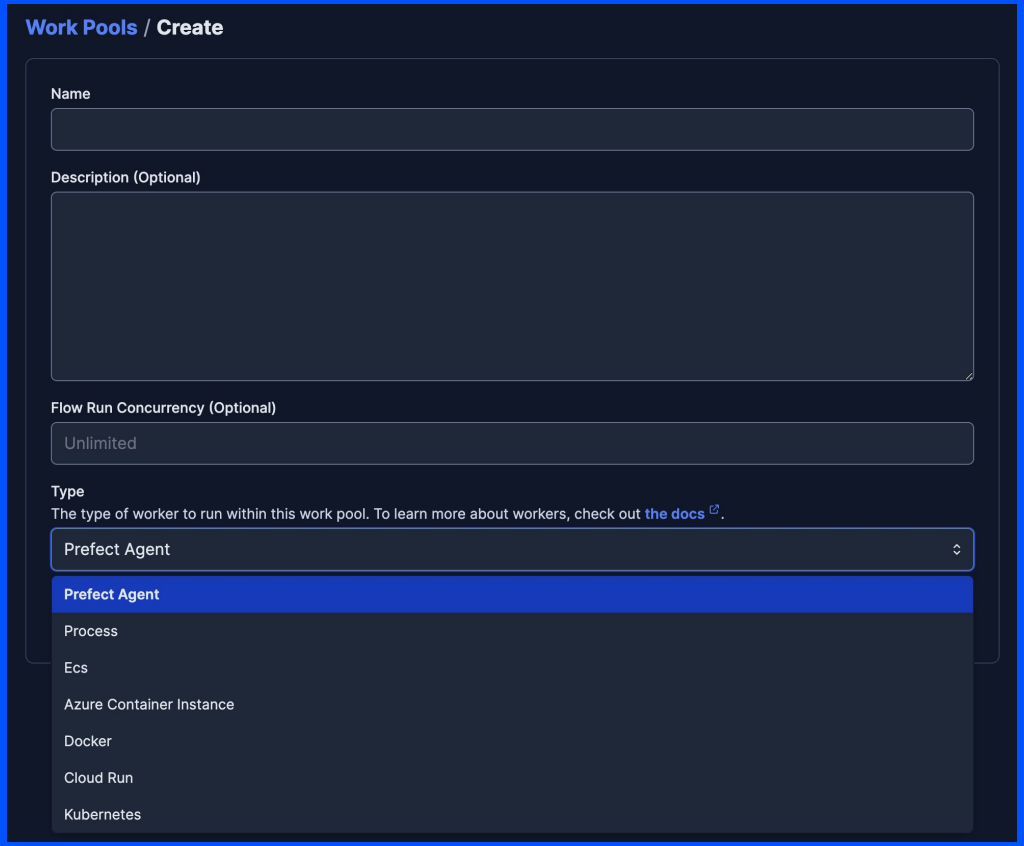
PREFECT ASSOCIATE
CERTIFICATION

Worker kicks off flow runs in the infrastructure type specified by worker & work pool

Work Pools - configure infrastructure

PREFECT ASSOCIATE CERTIFICATION

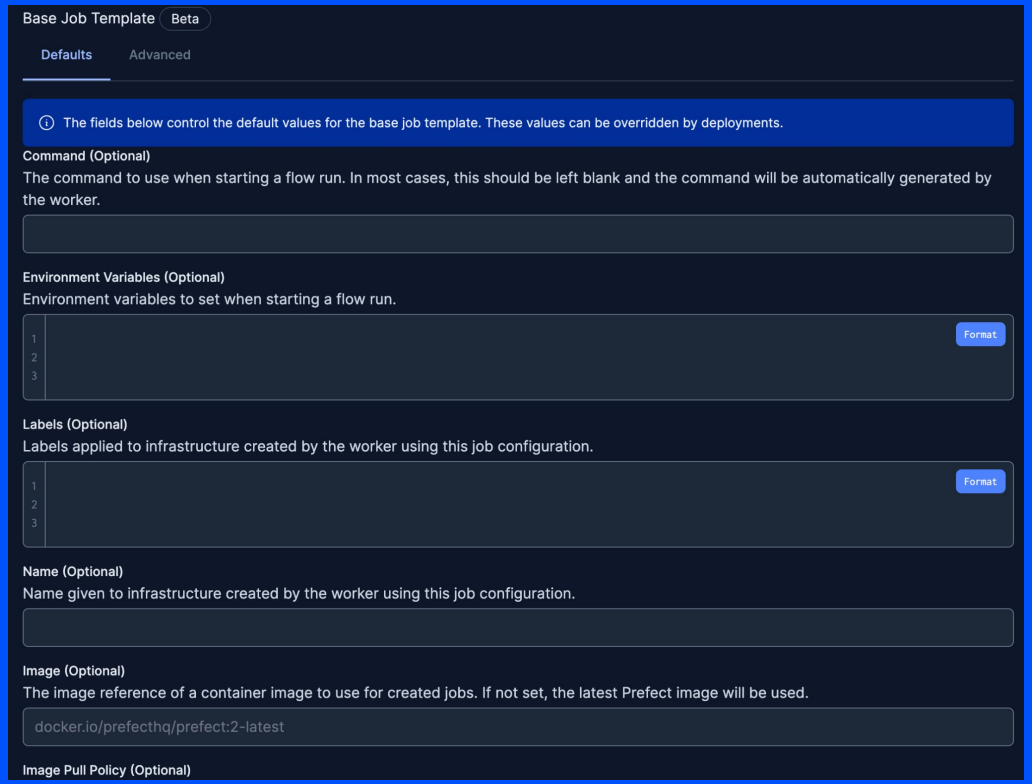
Choose Docker



Work pools

PREFECT ASSOCIATE CERTIFICATION

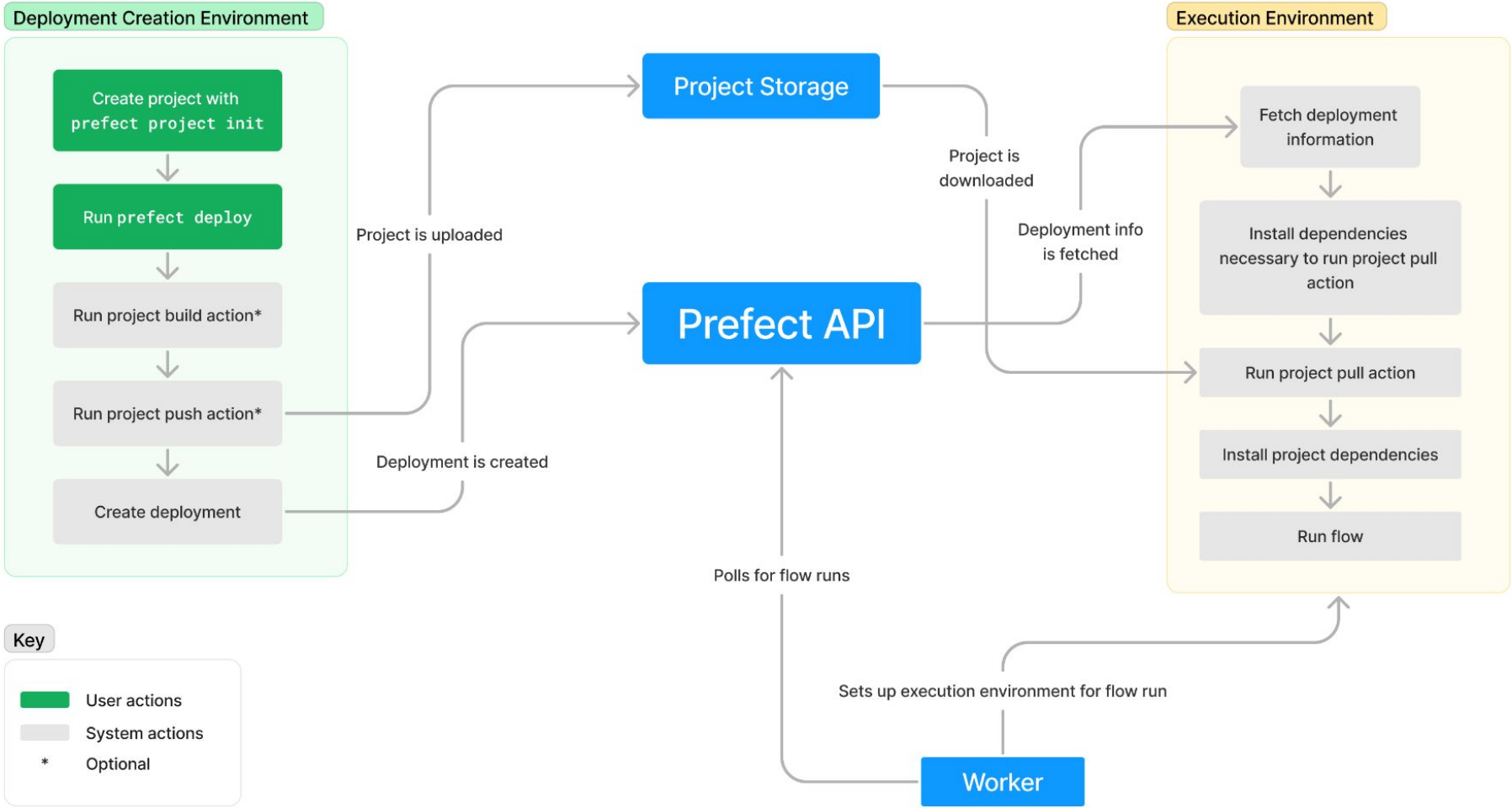
Can specify a particular image your team created



Work pools


Base Job Template
Beta
Defaults
Advanced
This is the JSON representation of the base job template. A work pool's job template controls infrastructure configuration for all flow runs in the work pool, and specifies the configuration that can be overridden by deployments.
For more information on the structure of a work pool's base job template, check out the docs.
Format
{
"job_configuration": {
"command": "{{ command }}",
"env": "{{ env }}",
"labels": "{{ labels }}",
"name": "{{ name }}",
"image": "{{ image }}",
"image_pull_policy": "{{ image_pull_policy }}",
"networks": "{{ networks }}",
"network_mode": "{{ network_mode }}",
"auto_remove": "{{ auto_remove }}",
"volumes": "{{ volumes }}",
"stream_output": "{{ stream_output }}",
"mem_limit": "{{ mem_limit }}",
"memswap_limit": "{{ memswap_limit }}",
"privileged": "{{ privileged }}"
},
"variables": {
"description": "Configuration class used by the Docker worker.\n\nAn instance of this class is passed to the Docker worker's `run`

Prefect Projects with Docker


Deployment Creation Environment
Create project with prefect project init
Run prefect deploy
Run project build action*
Run project push action*
Create deployment
Project Storage
Prefect API
Worker
Project is uploaded
Deployment is created
Project is downloaded
Deployment info is fetched
Polls for flow runs
Sets up execution environment for flow run
Execution Environment
Fetch deployment information
Install dependencies necessary to run project pull action
Run project pull action
Install project dependencies
Run flow
Key
User actions
System actions
* Optional

Docker

PREFECT ASSOCIATE
CERTIFICATION

With Prefect Projects you can

- Build a custom Docker image with flow code
- Push that image to a repo (or don't)
- Auto-include packages in *requirements.txt*

Prefect Project recipes

PREFECT ASSOCIATE CERTIFICATION

prefect project recipe ls

Name	Description
docker	Store project within a custom docker image alongside its runtime environment
docker-gcs	Store project within GCS and build a custom docker image for runtime
docker-git	Store project within a git repository and build a custom docker image for runtime
docker-s3	Store project within S3 and build a custom docker image for runtime
gcs	Store project within a GCS bucket
git	Store project within git repository
local	Store project on a local filesystem
s3	Store project within an S3 bucket

Prefect Project Docker Recipe

PREFECT ASSOCIATE
CERTIFICATION

```
prefect project init --recipe docker
```

Prompts for image name and tag if not provided

Required inputs for 'docker' recipe

Field Name	Description
image_name	The image name, including repository, to give the built Docker image
tag	The tag to give the built Docker image

Prefect Project Docker Recipe

PREFECT ASSOCIATE
CERTIFICATION

⚠️ *prefect project init* **will NOT** overwrite existing project files!

Docker - worker and work pool

PREFECT ASSOCIATE
CERTIFICATION

Start a Docker type worker to auto-create and connect to the pool named *docker-work*

```
prefect worker start -t docker -p docker-work
```

Docker - prefect.yaml


```yaml
name: '201'
prefect-version: 2.10.5

# build section allows you to manage and build docker images
build:
- prefect_docker.projects.steps.build_docker_image:
    requires: prefect-docker>0.1.0
    image_name: discdiver/demo
    tag: 0.0.1
    dockerfile: auto

# push section allows you to manage if and how this project is
push: null

# pull section allows you to provide instructions for cloning
pull:
- prefect.projects.steps.set_working_directory:
    directory: /opt/prefect/201
```

Docker - prefect.yaml

PREFECT ASSOCIATE
CERTIFICATION

If *dockerfile=Auto*, auto-creates Dockerfile when run *prefect deploy*

- Auto-discovers requirements.txt

```
FROM prefecthq/prefect:2.10.5-python3.10
COPY . /opt/prefect/201/
WORKDIR /opt/prefect/201/
RUN python -m pip install -r requirements.txt
```

Docker - prefect.yaml - with image push

PREFECT ASSOCIATE
CERTIFICATION

```yaml
name: '201'
prefect-version: 2.10.5

# build section allows you to manage and build docker images
build:
- prefect_docker.projects.steps.build_docker_image:
    requires: prefect-docker>0.1.0
    image_name: discdiver/demo
    tag: 0.0.1
    dockerfile: auto
    push: true

# push section allows you to manage if and how this project is
push: null

# pull section allows you to provide instructions for cloning
pull:
- prefect.projects.steps.set_working_directory:
    directory: /opt/prefect/201
```

prefect deploy with Docker

PREFECT ASSOCIATE
CERTIFICATION

Builds image

If *build->push: True*, then pushes image to registry

Flow run

Docker configuration in the work pool can be overridden by a deployment

(set in deployment.yaml)

Docker - deployment.yaml

PREFECT ASSOCIATE
CERTIFICATION

```yaml
deployments:
- name: null
  version: null
  tags: []
  description: null
  schedule: {}
  flow_name: null
  entrypoint: null
  parameters: {}
  work_pool:
    job_variables:
      image: '{{ image_name }}'
```

Docker

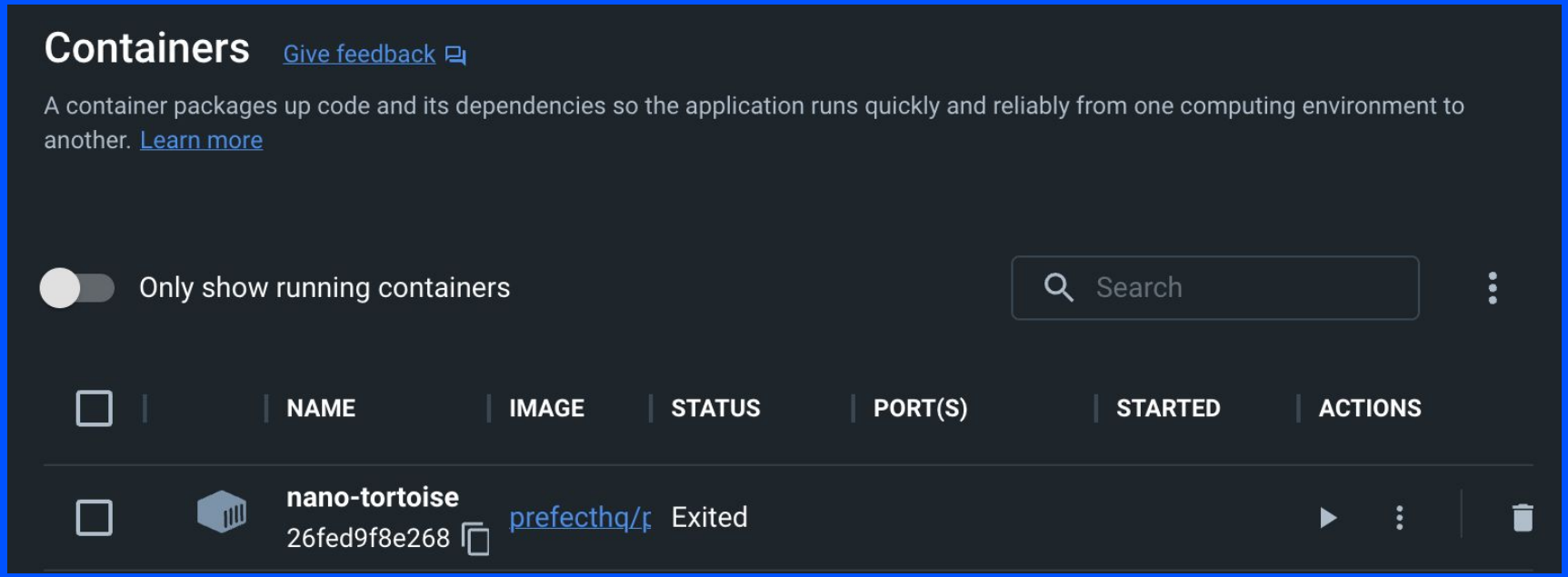
PREFECT ASSOCIATE
CERTIFICATION

- *prefect deploy*
- Run deployment
- 

Infrastructure - Docker Container

PREFECT ASSOCIATE
CERTIFICATION

Check out Docker Desktop



201 Recap

You've seen how to use Docker with projects, workers & work pools!

PREFECT

Bonus: Caching

Caching

PREFECT ASSOCIATE
CERTIFICATION

What?

Why?

⚠️ task only

Requires persisting results

Caching: cache_key_fn

PREFECT ASSOCIATE
CERTIFICATION

`@task(cache_key_fn=task_input_hash)`

```python
from prefect import flow, task
from prefect.tasks import task_input_hash

@task(cache_key_fn=task_input_hash)
def hello_task(name_input):
    print(f"Hello {name_input}!")

@flow
def hello_flow(name_input):
    hello_task(name_input)
```

Caching

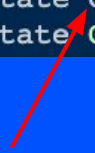
PREFECT ASSOCIATE CERTIFICATION

First run

```
22:32:04.227 | INFO    | prefect.engine - Created flow run 'smoky-hippo' for flow 'hello-flow'
22:32:04.311 | INFO    | Flow run 'smoky-hippo' - Created task run 'hello_task-0' for task 'hello_task'
22:32:04.311 | INFO    | Flow run 'smoky-hippo' - Executing 'hello_task-0' immediately...
Hello Liz!
22:32:04.353 | INFO    | Task run 'hello_task-0' - Finished in state Completed()
22:32:04.368 | INFO    | Flow run 'smoky-hippo' - Finished in state Completed('All states completed.')
```

Second run

```
22:33:02.606 | INFO    | prefect.engine - Created flow run 'able-scallop' for flow 'hello-flow'
22:33:02.701 | INFO    | Flow run 'able-scallop' - Created task run 'hello_task-0' for task 'hello_task'
22:33:02.702 | INFO    | Flow run 'able-scallop' - Executing 'hello_task-0' immediately...
22:33:02.720 | INFO    | Task run 'hello_task-0' - Finished in state Cached(type=COMPLETED)
22:33:02.735 | INFO    | Flow run 'able-scallop' - Finished in state Completed('All states completed.')
```



Caching

PREFECT ASSOCIATE
CERTIFICATION

What will happen if you run with a different argument?

```
22:36:07.750 | INFO    | prefect.engine - Created flow run 'daffodil-millipede' for flow 'hello-flow'
22:36:07.836 | INFO    | Flow run 'daffodil-millipede' - Created task run 'hello_task-0' for task 'hello_task'
22:36:07.837 | INFO    | Flow run 'daffodil-millipede' - Executing 'hello_task-0' immediately...
Hello Marvin!
22:36:07.877 | INFO    | Task run 'hello_task-0' - Finished in state Completed()
22:36:07.892 | INFO    | Flow run 'daffodil-millipede' - Finished in state Completed('All states completed.')
```

Caching: cache_expiration ⏳

PREFECT ASSOCIATE
CERTIFICATION

```python
from prefect import flow, task
from prefect.tasks import task_input_hash
from datetime import timedelta

@task(cache_key_fn=task_input_hash, cache_expiration=timedelta(minutes=1))
def hello_task(name_input):
    print(f"Hello {name_input}!")

@flow
def hello_flow(name_input):
    hello_task(name_input)
```

PREFECT

AMA

PREFECT

Wrap

Brief feedback survey

PREFECT ASSOCIATE
CERTIFICATION

Please let us know what went well and what could be improved. 

Prefect Practitioner Certification Course

Congratulations!