



PREFECT ASSOCIATE
CERTIFICATION



Introductions

Introductions



PREFECT ASSOCIATE
CERTIFICATION

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



Overview

Prefect provides tools for working
with complex systems

so you can stop wondering about
your workflows



If you give an engineer a job...

- 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 Technologies, Inc.
prod-releases

- Flow Runs
- Flows
- Work Pools
- Blocks
- Variables
- Automations
- Task Run Concurrency
- Event Feed
- Artifacts

Workspace Sharing
Workspace Settings

Flow Runs / sweet-mink

Completed 2023/05/04 11:55:51 AM 31s 7 task runs

Flow release-package

Logs Task Runs Subflow Runs Results Artifacts Details Parameters

Level: all Oldest to newest

May 4th, 2023

- INFO Created task run 'Get directory-0' for task 'Get directory'
- INFO Submitted task run 'Get directory-0' for execution.
- INFO Created task run 'Clone repo-0' for task 'Clone repo'
- INFO Submitted task run 'Clone repo-0' for execution.

11:55:51 AM prefect.flow_runs
11:55:51 AM prefect.flow_runs
11:55:52 AM prefect.flow_runs
11:55:52 AM prefect.flow_runs
11:55:53 AM prefect.flow_runs

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

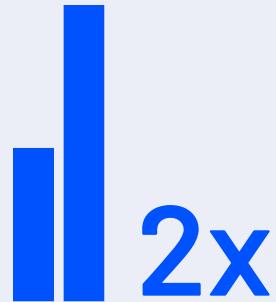
- UI
- Notifications
- Event feed
- Collaboration



With extraordinary impacts.



of users reported much faster development cycles



average development productivity boost



reduction in pipeline errors

SOURCE: Prefect community survey, November 2021.



Goals



Goals

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



Norms

Zoom

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

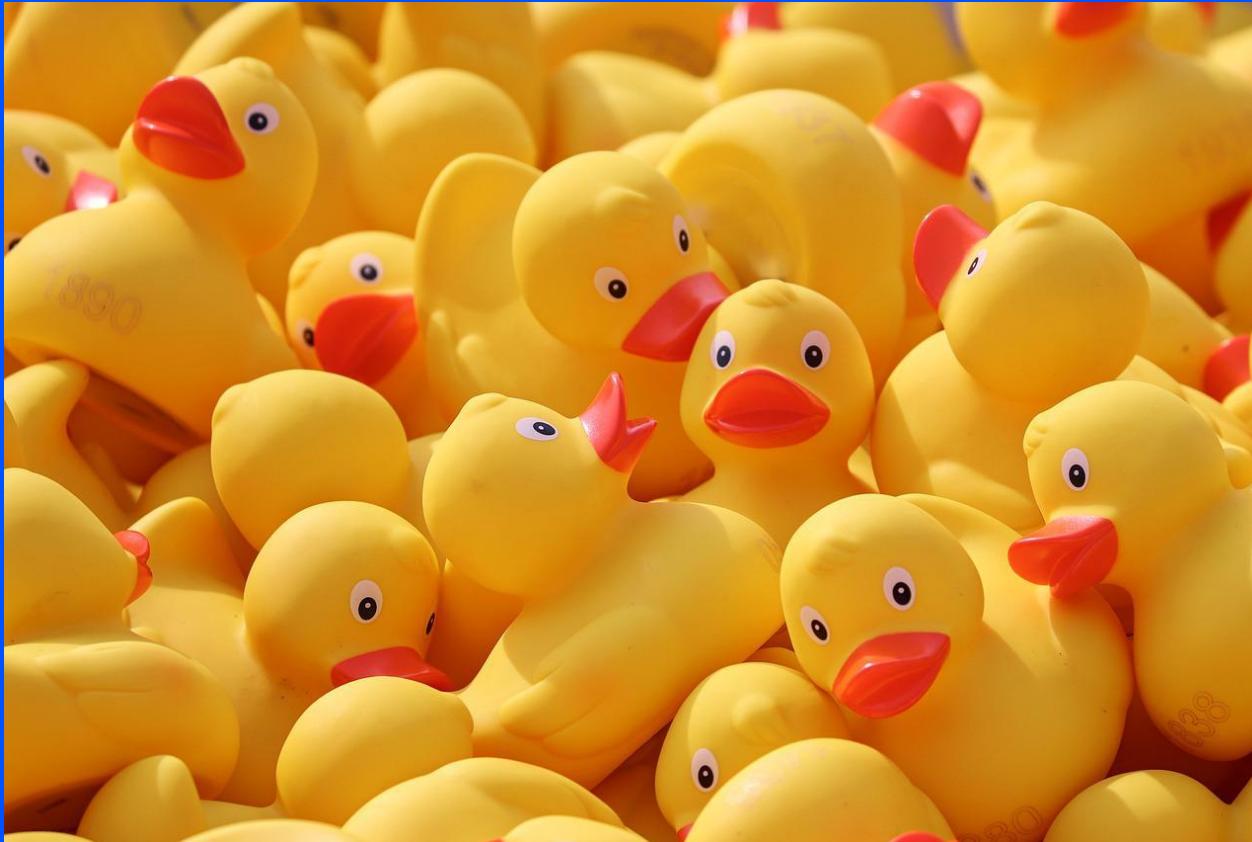
Slack

- Use threads
- Emoji responses 😊

Welcome to the land of the ducks



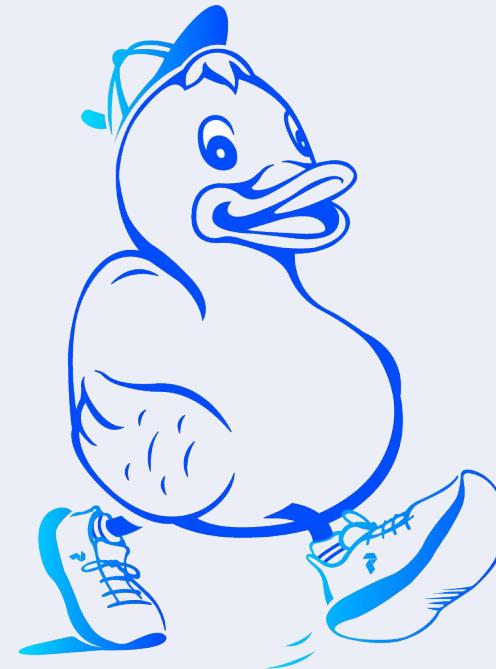
PREFECT ASSOCIATE
CERTIFICATION



Meet Minerva

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

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



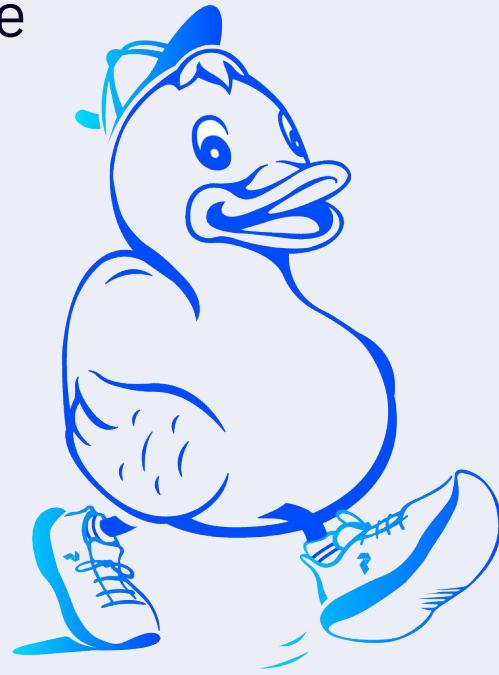


PREFECT ASSOCIATE
CERTIFICATION

Your charge

Help Minerva build data pipelines that are

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



MODULE

101 - Prefect basics

101 Agenda

- From Python function to Prefect flow
- Tasks
- Docs

Initial project

Help Minerva get some weather data. 

open-meteo.com/en

Fetch data: Basic Python function



```
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

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



Make it a flow



PREFECT ASSOCIATE
CERTIFICATION

```
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



PREFECT ASSOCIATE
CERTIFICATION

```
17:32:35.177 | INFO      | prefect.engine - Created flow run 'observant-chihuahua' for flow 'fetch-weather'  
Most recent temp C: 15.2 degrees  
17:32:36.738 | INFO      | Flow run 'observant-chihuahua' - Finished in state Completed()
```



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

```
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



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

Pipeline (assembly) function



```
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

```
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

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

Pipeline function flow



Pass the result of one task to another inside a flow

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

Logs from run



PREFECT ASSOCIATE
CERTIFICATION

```
17:34:15.961 | INFO    | prefect.engine - Created flow run 'rapid-smilodon' for flow 'pipeline'
17:34:16.735 | INFO    | Flow run 'rapid-smilodon' - Created task run 'fetch_weather-0' for task 'fetch_weather'
17:34:16.736 | INFO    | Flow run 'rapid-smilodon' - Executing 'fetch_weather-0' immediately...
17:34:17.820 | INFO    | Task run 'fetch_weather-0' - Finished in state Completed()
17:34:17.941 | INFO    | Flow run 'rapid-smilodon' - Created task run 'save_weather-0' for task 'save_weather'
17:34:17.942 | INFO    | Flow run 'rapid-smilodon' - Executing 'save_weather-0' immediately...
17:34:18.389 | INFO    | Task run 'save_weather-0' - Finished in state Completed()
17:34:18.513 | INFO    | Flow run 'rapid-smilodon' - Finished in state Completed()
```

Tasks dos and don'ts

-  Do call tasks within a flow
-  Don't call tasks from other tasks
-  Do keep tasks small



Welcome to Prefect

Prefect enables you to build and observe resilient data workflows so that you can understand, react to, and recover from unexpected changes. It's the easiest way to transform any Python function into a unit of work that can be observed and orchestrated. Just bring your Python code, sprinkle in a few decorators, and go!

With Prefect you gain:

- scheduling
- retries
- logging
- caching
- async
- notifications
- observability

Trying to implement these features from scratch is a huge pain that takes time, headaches, and money. That's why Prefect offers all this functionality and more!

The screenshot shows the Prefect Cloud interface. On the left is a sidebar with navigation links: Prefect Docs, Getting Started (selected), Installation, Cloud Quickstart, Tutorial, Concepts, Cloud, Host, Guides, Integrations, API References, and Community. The main area has a title "Welcome to Prefect" and a paragraph about Prefect's benefits. Below that is a section titled "With Prefect you gain:" with a bulleted list. At the bottom is a note about implementing features from scratch. On the right is a detailed view of a "Flow Runs / sweet-mink" run. The run status is "Completed" at 2023/05/04 11:55:51 AM, took 31s, and had 7 task runs. The timeline shows tasks running sequentially: "Get directory-0" at 11:55:50 AM, "Install dependencies-0" at 11:55:55 AM, "Close connection-0" at 11:56:00 AM, "Publish latest version-0" at 11:56:05 AM, "Release-0" at 11:56:10 AM, "Cleanup-0" at 11:56:15 AM, and "Send notifications-0" at 11:56:20 AM. Arrows indicate the flow of tasks and dependencies between them.



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 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

The screenshot shows the GitHub repository page for `PrefectHQ/prefect`. The page has a dark theme. At the top, there are buttons for `Edit Pins`, `Unwatch` (with 153 notifications), `Fork` (with 1.1k forks), `Starred` (with 11.3k stars), and a dropdown menu. Below the header, there are tabs for `Code` (selected), `Issues` (671), `Pull requests` (53), `Discussions`, `Actions`, `Projects`, and a `...` button. In the main content area, there's a navigation bar with `main`, `Go to file`, `Add file`, `<> Code` (highlighted in green), and `About`. A user notification at the bottom left says `peytonrunyan and madkinsz Fix er...` was posted 13 hours ago with 13,871 reactions. The `About` section on the right describes the repository as "The easiest way to coordinate your dataflow".



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

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

MODULE

102 - Intro to orchestration

102 Agenda

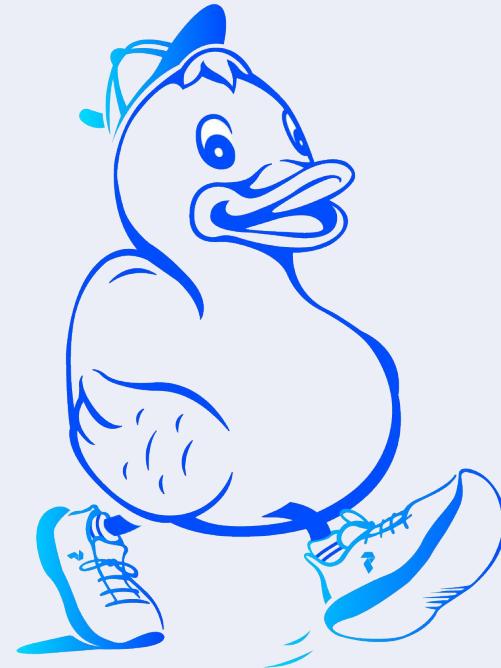
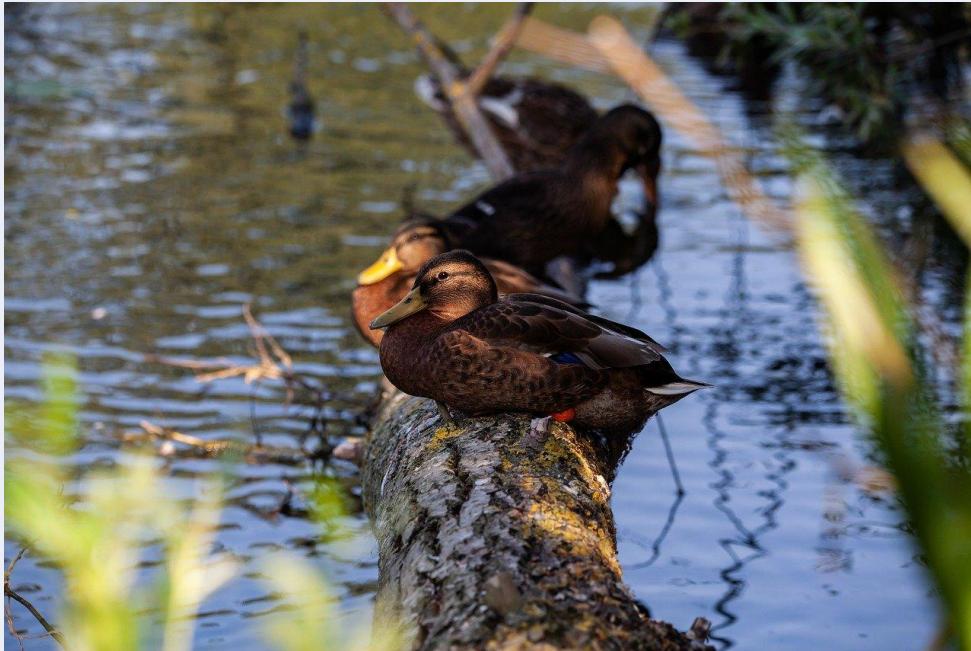


PREFECT ASSOCIATE
CERTIFICATION

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

Logging

Ducks on logs:



log_prints



PREFECT ASSOCIATE
CERTIFICATION

Log the print statements

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

Logging

Create custom logs with `get_run_logger`

```
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-example-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



```
@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

```
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' when 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



```
@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

```
Exception
22:30:35.032 | INFO    | Task run 'fetch_cat_fact-0' - Received non-final state 'AwaitingRetry' when 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-finches' - Finished in state Completed('All states complete d.')
```



Prefect server



Start a Prefect server locally



Open another terminal window

Run:

```
prefect server start
```

The Prefect CLI

Start commands with `prefect`

`--help` is always available



prefect --help



Options

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

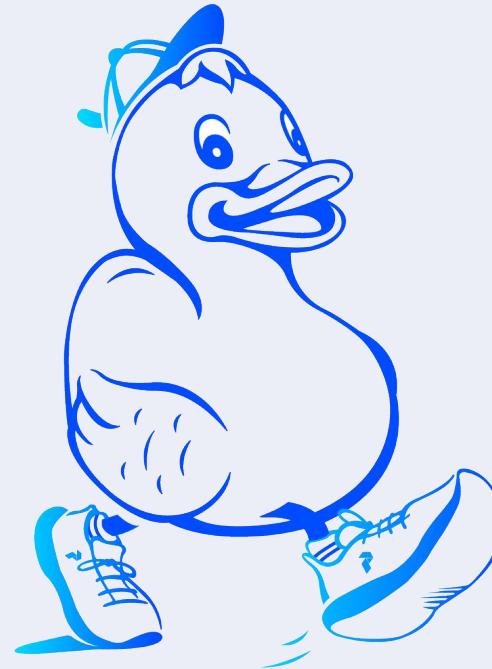
Commands

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



Fire up the Prefect Server



Start a Prefect server locally



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

Flow Runs

Flows

Work Pools

Blocks

(x) Variables

Notifications

Task Run Concurrency

Artifacts

Flow Runs

Date Range: 05/02/2023 > 05/10/2023

States: All run states

Flows: All flows

Deployments: All deployments

Work Pools: All pools

Tags: All tags

Timeline: 1s, 0.75s, 0.50s, 0.25s, 0s

Tue 02 Wed 03 Thu 04 Fri 05 Sat 06 May 07 Mon 08 Tue 09 Wed 10

1 Flow run

Search by run name

Newest to oldest

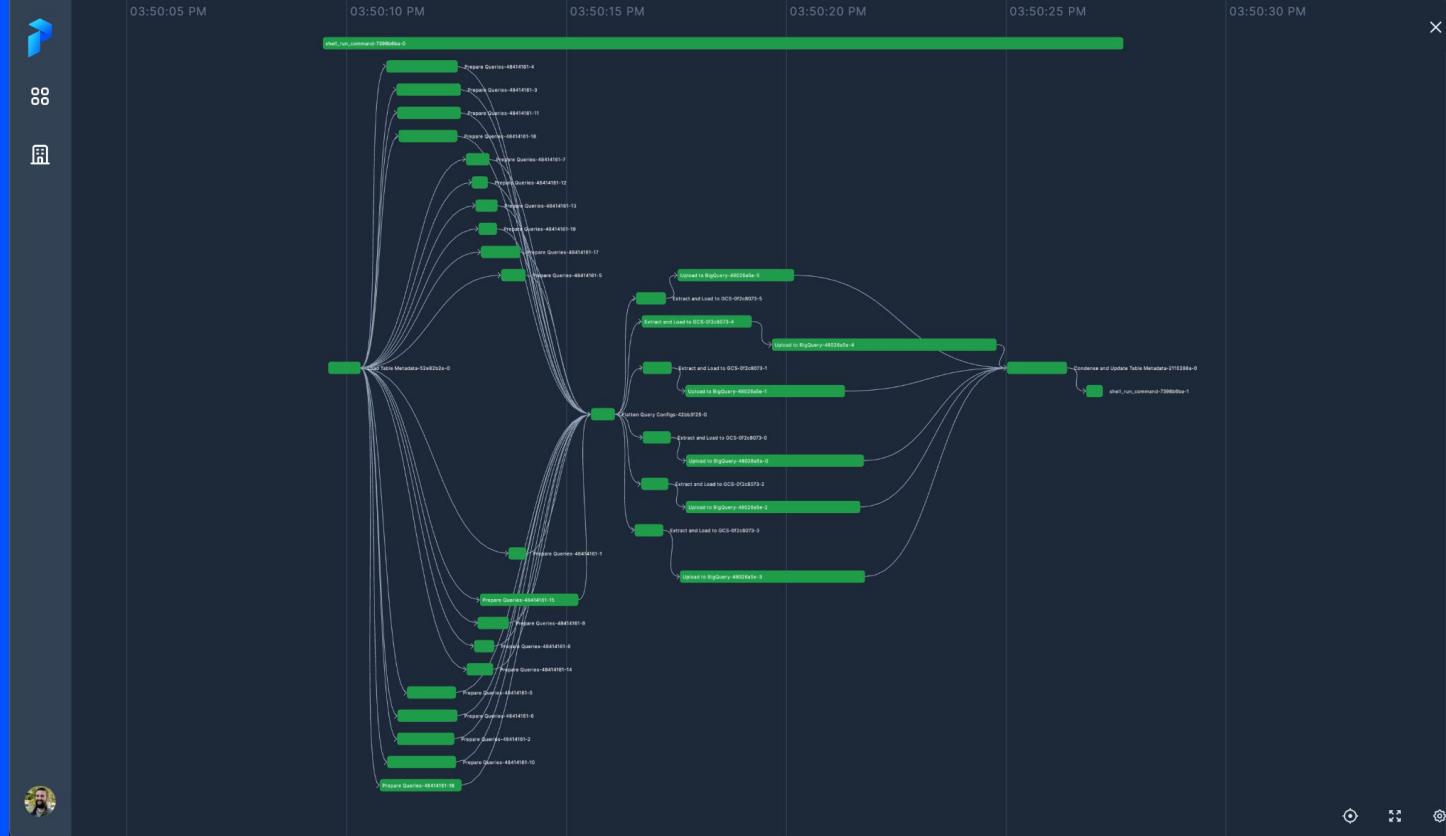
fetch-weather > righteous-aardwolf

Completed 2023/05/09 01:41:05 PM 1s None

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



PREFECT ASSOCIATE
CERTIFICATION

The data returned by a flow or a task

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

1 is the result

Results



PREFECT ASSOCIATE
CERTIFICATION

```
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

The screenshot shows the Prefect UI interface with the "Results" tab selected. The top navigation bar includes tabs for Logs, Results (highlighted with a blue border), Artifacts, Task Runs, Subflow Runs, and Parameters. Below the navigation bar, there are two main sections: "Flow run" and "Task runs".

Flow run

RESULT
Apr 13th, 2023 at 12:09 PM Created

Task runs

RESULT
`my_task-0`

Result of type `DataFrame` persisted to:
`/Users/jeffhale/.prefect/storage/c65d28dcc374424ba7212a39dd19418b`

Results



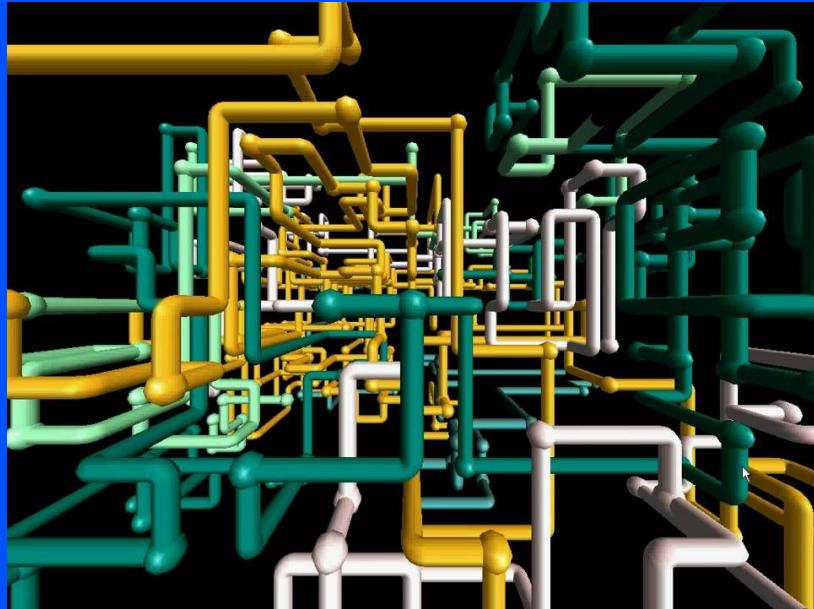
PREFECT ASSOCIATE
CERTIFICATION

The result in the storage folder

```
✓ .PREFECT
  ✓ storage
    {} c65d28dcc374424ba7212a39dd19418b
      storage > {} c65d28dcc374424ba7212a39dd19418b > ...
        1   {"serializer": {"type": "pickle", "picklelib": "cloudpickle", "picklelib_version": "2.2.1"},}
        2   "data": "gAWVxwIAAAAAACMEXBhbmRhcy5jb3JlLmZyYW1llIwJRGF0YUZyYW1llJ0UKYGuFZQojARfbWdy\\nIwec"
        3   "prefect_version": "2.10.3"
        4
        5
        6
```

Subflows

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



Subflows



```
@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"\n🐱: {cat_fact} \n🐶: {dog_fact}")
```

Subflows



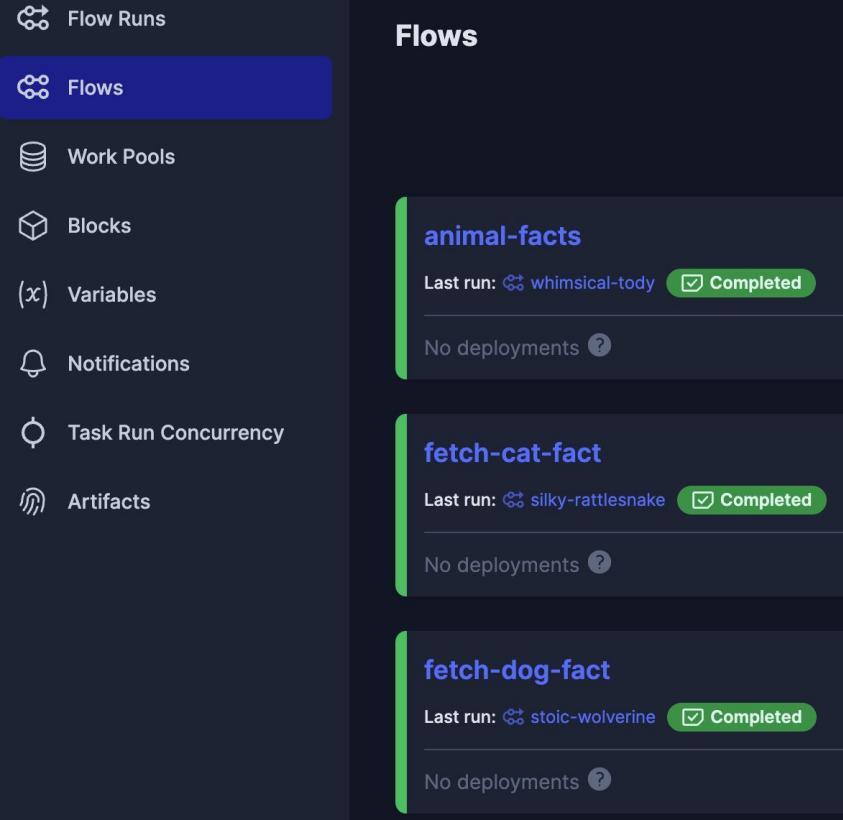
PREFECT ASSOCIATE
CERTIFICATION

```
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



Subflows in UI



The screenshot shows the Prefect UI interface. On the left, a sidebar menu lists several options: Flow Runs, Flows (which is selected and highlighted in blue), Work Pools, Blocks, Variables, Notifications, Task Run Concurrency, and Artifacts. The main area is titled "Flows" and displays three completed subflows:

- animal-facts**: Last run: `whimsical-tody` (Completed)
- fetch-cat-fact**: Last run: `silky-rattlesnake` (Completed)
- fetch-dog-fact**: Last run: `stoic-wolverine` (Completed)

Each subflow entry includes a "No deployments" link.

Timeline view



PREFECT ASSOCIATE
CERTIFICATION

Flow Runs

Flows

Work Pools

Blocks

Variables

Notifications

Task Run Concurrency

Artifacts

Flow Runs / whimsical-tody

Completed | 2023/05/09 02:16:09 PM | 1s | None

Flow `animal-facts`

	02:15:30 PM	02:16:00 PM	02:16:30 PM	02:17:00 PM
			 fetch-cat-fact / silky-rattlesnake	
			 fetch-dog-fact / stoic-wolverine	

Logs Task Runs Subflow Runs Results Artifacts Details Parameters

Level: all | Oldest to newest

May 9th, 2023

INFO Created subflow run 'silky-rattlesnake' for flow 'fetch-cat-fact' 02:16:09 PM prefect.flow_runs

INFO Created subflow run 'stoic-wolverine' for flow 'fetch-dog-fact' 02:16:10 PM prefect.flow_runs

INFO 🐱: A cat lover is called an Ailurophilia (Greek: cat-lover). 02:16:10 PM prefect.flow_runs

INFO 🐕: Wow, check out those choppers! Puppies have 28 teeth and normal adult dogs have 42. 02:16:10 PM prefect.flow_runs

INFO Finished in state Completed('All states completed.') 02:16:10 PM prefect.flow_runs

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
INFO    🐶: 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
```



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

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



Lab 102

Lab 102

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

MODULE

103 - Cloud & Blocks

103 Agenda



PREFECT ASSOCIATE
CERTIFICATION

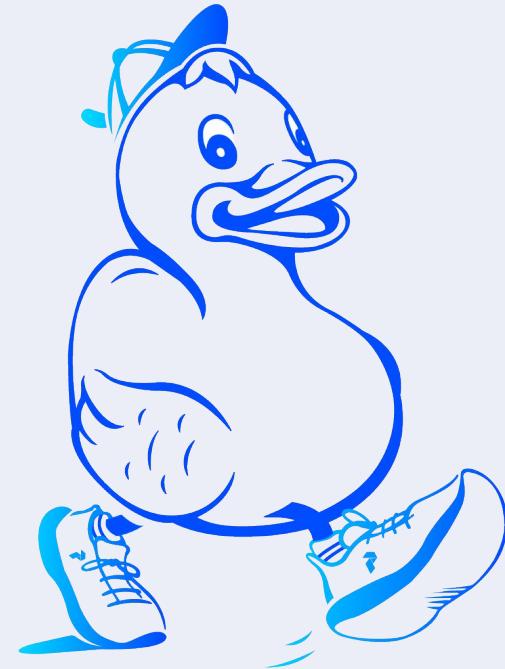
- Cloud
- Hybrid Model
- Intro to Blocks 
- Automations 



PREFECT ASSOCIATE
CERTIFICATION

To the cloud

Like other ducks, Minerva is into clouds.





Go to app.prefect.cloud in browser

- Sign up or sign in



Authenticate CLI through browser or API key

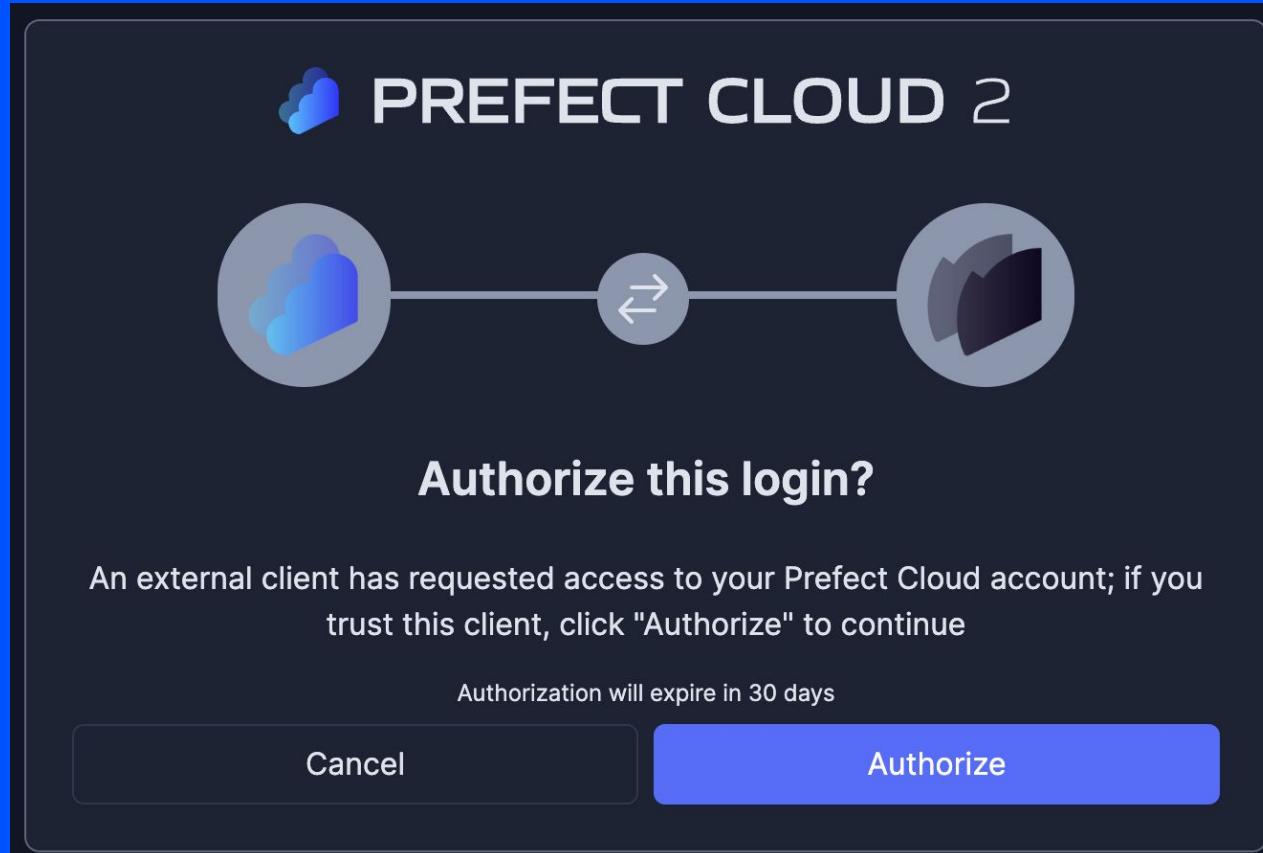
Run: *prefect cloud login*

```
? 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





Or paste an API key

Manually grab an API key from Prefect Cloud in the browser

Prefect Cloud - API Key



PREFECT ASSOCIATE
CERTIFICATION

The screenshot shows the Prefect Cloud interface with a dark theme. On the left, there's a sidebar with icons for Workspaces, Projects, and Data. The main area is titled "Workspaces" and shows a workspace named "data-time" created by "jeffgmai" (1 member). Below the workspace list, there's a toggle switch for light/dark mode, followed by a user profile for "Jeff Hale". A red arrow points from the bottom-left towards this profile. Below the profile, there's a menu with links to "Prefect Docs", "Prefect Cloud REST API Docs", "Community Forum", "Community Slack", and "Sign Out". Another red arrow points from the bottom-left towards the "Sign Out" link.

Prefect Cloud - API Key



PREFECT ASSOCIATE
CERTIFICATION

Jeff Hale

Profile

API Keys

Billing

Preferences

API Keys

+

4 API keys

Search API keys

Name	Created	Expiration
cli-e5c61a7f-0232-413d-8353-320e4ffc9311	Apr 13th, 2023	2023/05/13 12:00:00 AM
cli-d207f44e-ec61-4943-9ad6-f9e1bd727875	Apr 19th, 2023	2023/05/19 12:00:00 AM
demos	Aug 28th, 2022	Never
jkey	Jan 7th, 2023	Never

Prefect Cloud - API Key



PREFECT ASSOCIATE
CERTIFICATION

Name	Created
Create API Key X	
Name	
Expiration Date	06/08/2023 edit
<input type="checkbox"/> Never Expire	
Cancel	Create



PREFECT

Profiles

Prefect profiles - list

```
prefect profile ls
```

Available Profiles:

```
* default
    local
jeffmshale
    gh2
prefect-more
```



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 Cloud



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

Prefect Cloud Workspaces



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





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

Prefect Cloud - Organization



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

Prefect Cloud - Enterprise



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

Prefect Cloud



PREFECT ASSOCIATE
CERTIFICATION

Personal

Start with managed orchestration.

**Free
Forever**

[CREATE YOUR WORKSPACE](#)

3 free Users

1 free Workspace

7 day history

Organization

Prefect for scaling out with your team. Get a free 14 day trial.

Starts at

\$450/mo

[START YOUR FREE TRIAL](#)

Read-Only Users

Service Accounts + Enforced SSO

30 day history

Enterprise

Security, compliance and custom configurations.

Get an estimate

[TALK TO SALES](#)

Enterprise support

SCIM + Custom RBAC

Custom configurations

Dedicated Cloud

Dedicated Instance of Prefect Cloud

Get in touch

[TALK TO A PLATFORM ENGINEER](#)

No Resource Limits

Performance x Security

Implementation and Extended SLA

Organization level

- Admin
- Member

Workspace level

- Viewer
- Runner
- Developer
- Owner



PREFECT

Hybrid Model



PREFECT HYBRID MODEL

YOUR EXECUTION ENVIRONMENT

WORKER

Kicks off flow runs

YOUR FLOW CODE

Opens connection

Returns scheduled flow runs

Sends metadata

PREFECT CLOUD

API & DATABASE

UI

Displays

INTEGRATIONS



+ MANY MORE

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

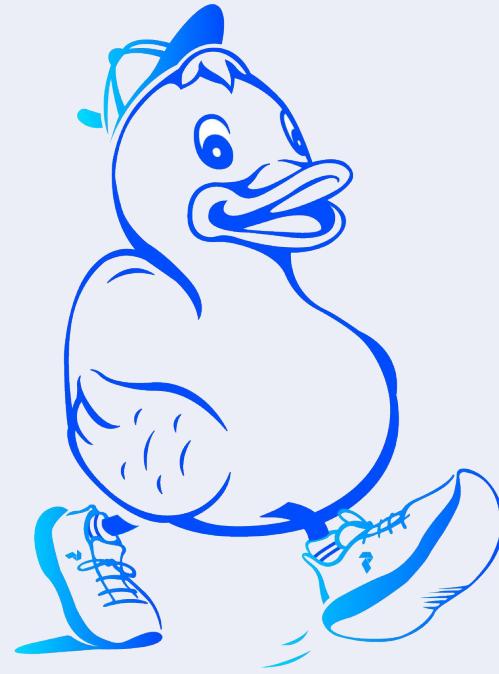
Blocks



PREFECT ASSOCIATE
CERTIFICATION

Blocks

Blocks are a cool Prefect feature

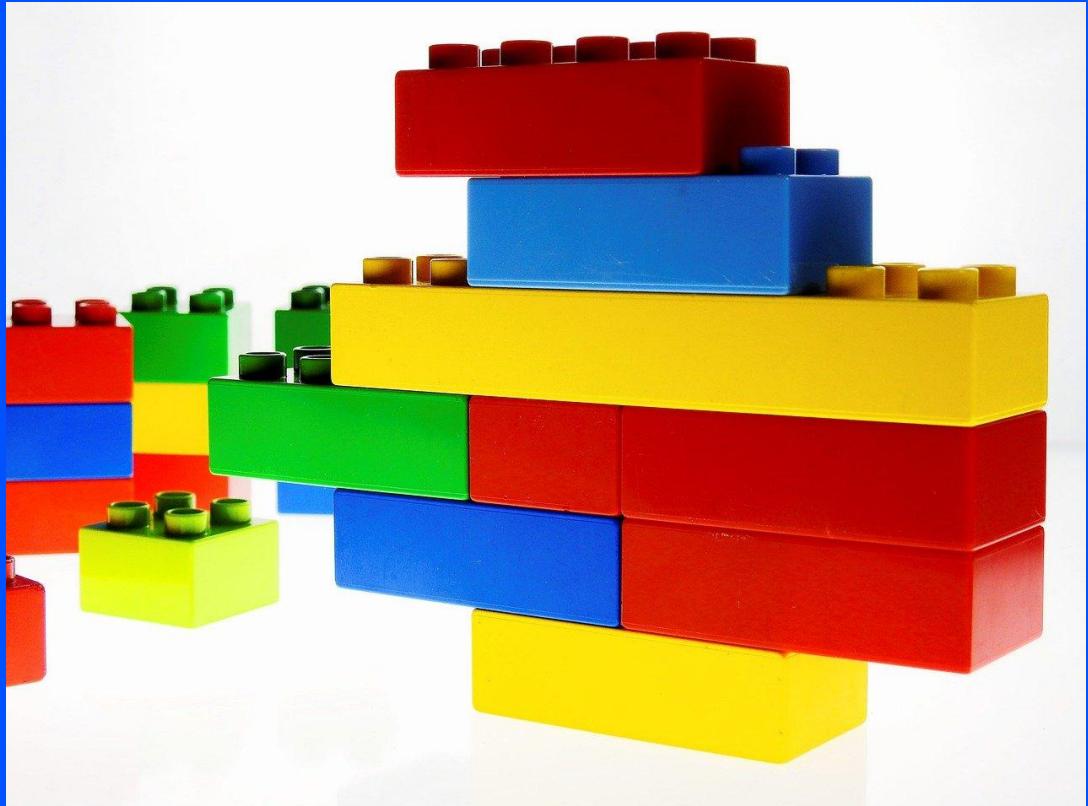


Blocks

Configuration

+

Code



Blocks in UI



PREFECT ASSOCIATE
CERTIFICATION



Blocks +

May 9th, 2023 01:04 AM May 9th, 2023 11:59 PM

48 Blocks

Search blocks Capability: all Type: all

Name	Capabilities
AWS Credentials / my-aws-creds-block	get-directory put-directory read-path write-path
Azure / azure-demo	get-directory put-directory read-path write-path

Select a Secret block



PREFECT ASSOCIATE
CERTIFICATION

 Remote File System Store data as a file on a remote file system. Supports any remote file system supported by 'fsspec'. The file system is specified using a protocol. For example, "s3://my-...." <code>get-directory</code> <code>put-directory</code> <code>read-path</code> <code>write-path</code> Add +	 S3 Store data as a file on AWS S3. <code>get-directory</code> <code>put-directory</code> <code>read-path</code> <code>write-path</code> Add +	 S3 Bucket Block used to store data using AWS S3 or S3-compatible object storage like MinIO. This block is part of the prefect-aws collection. Install prefect-aws with 'pip install...' <code>get-directory</code> <code>put-directory</code> <code>read-path</code> <code>write-path</code> Add +
 Secret A block that represents a secret value. The value stored in this block will be obfuscated when this block is logged or shown in the UI. Add +	 Shell Operation A block representing a shell operation, containing multiple commands. For long-lasting operations, use the trigger method and utilize the block as a context... Add +	 Slack Credentials Block holding Slack credentials for use in tasks and flows. This block is part of the prefect-slack collection. Install prefect-slack with 'pip install prefect-slack' to use this block. Add +
 Slack Incoming Webhook	 Slack Webhook	 SMB

Create a Secret block



PREFECT ASSOCIATE
CERTIFICATION

Blocks / Choose a Block / Secret / Create

Block Name

so-secret

Value

A string value that should be kept secret.

.....

Cancel

Create



Secret

A block that represents a secret value. The value stored in this block will be obfuscated when this block is logged or shown in the UI.

Use the Secret block



PREFECT ASSOCIATE
CERTIFICATION

Blocks / so-secret

May 9th, 2023 12:00 AM May 9th, 2023 11:59 PM

Paste this snippet **into your flows** to use this block. Need help? [View Docs](#)

```
from prefect.blocks.system import Secret

secret_block = Secret.load("so-secret")

# Access the stored secret
secret_block.get()
```

Value

Secret

A block that represents a secret value. The value stored in this block will be obfuscated when this block is logged or shown in the UI.

Create a JSON block from code



```
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



Reusable, modular, configuration + code

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



Cloud Features

Automations



PREFECT ASSOCIATE
CERTIFICATION

jeffgmail
data-time

Flow Runs

Flows

Work Pools

Blocks

Variables

→

Automations

→

Task Run Concurrency

Event Feed

Artifacts

Flow Runs

Date Range

05/02/2023 > 05/10/2023



Flows

All flows

Deployments

All deployments





Automations

Automations



PREFECT ASSOCIATE
CERTIFICATION

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

jeffgmail
data-time

- Flow Runs
- Flows
- Work Pools
- Blocks
- (x) Variables

Automations

Automations +

email-jeff-cat

Email Jeff when cat fact fails.

Action Send a notification

⋮

Automations



PREFECT ASSOCIATE
CERTIFICATION

Automations / Create Documentation 

01 Trigger 02 Actions 03 Details

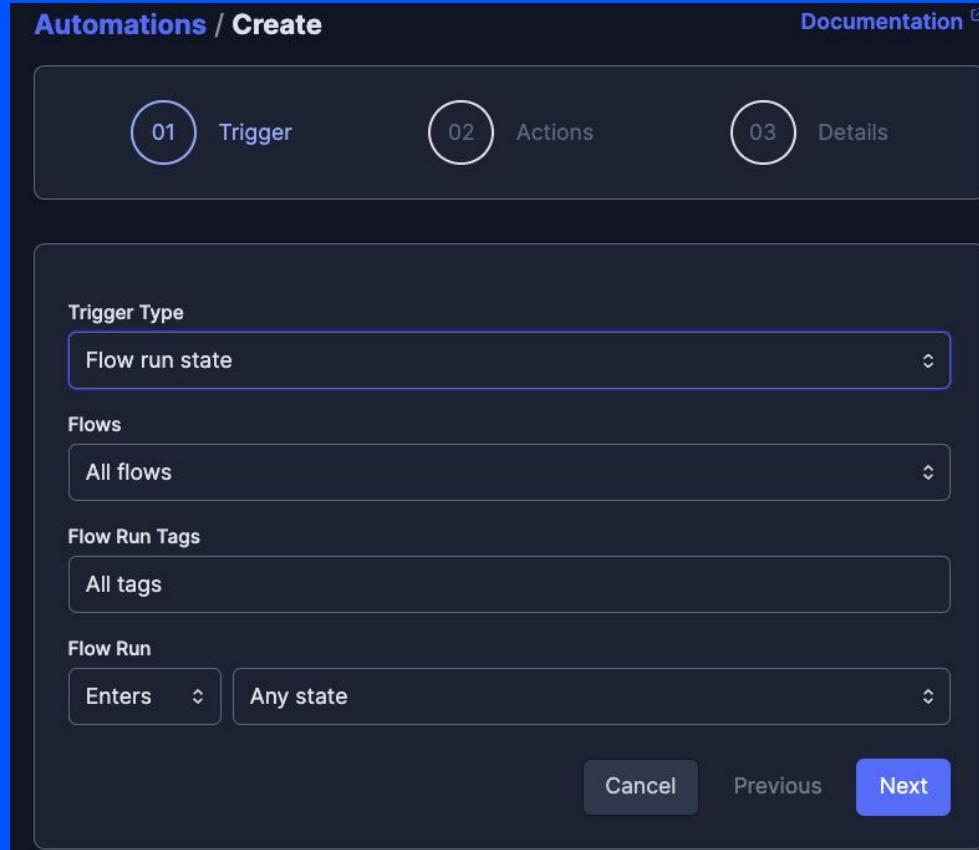
Trigger Type
Flow run state 

Flows
All flows 

Flow Run Tags
All tags 

Flow Run
Enters  Any state 

Buttons
Cancel Previous Next



Automations



PREFECT ASSOCIATE
CERTIFICATION

Automations / Create Documentation

Trigger 02 Actions 03 Details

Action 1

Action Type

Send a notification

Block

Add +

Subject

Prefect flow run notification

Body

```
Flow run {{ flow.name }}/{{ flow_run.name }} entered state `{{ flow_run.state.name }}` at {{ flow_run.state.timestamp }}.  
Flow ID: {{ flow_run.flow_id }}  
Flow run ID: {{ flow_run.id }}  
Flow run URL: {{ flow_run|ui_url }}
```

Automations



PREFECT ASSOCIATE
CERTIFICATION

Blocks / Choose a Block

If you don't see a block for the service you're using, check out our [Collections Catalog](#) to view a list of integrations and their corresponding blocks.

8 Blocks

Search blocks

Capability: notify



Email

Block that allows an email to be sent to a list of email addresses via Sendgrid. This block is only available for use within automations...

notify

Add +



Mattermost Webhook

Enables sending notifications via a provided Mattermost webhook.

notify

Add +



Microsoft Teams Webhook

Enables sending notifications via a provided Microsoft Teams webhook.

notify

Add +



Opsgenie Webhook

Enables sending notifications via a provided Opsgenie webhook.

notify

Add +

Automations



PREFECT ASSOCIATE
CERTIFICATION

Blocks / Choose a Block / Email / Create

Block Name

Emails

List of email addresses to send the email to

▼

Cancel

Create



Email

Block that allows an email to be sent to a list of email addresses via Sendgrid. This block is only available for use within automations and cannot be...

notify



PREFECT

Events

Events



PREFECT ASSOCIATE
CERTIFICATION

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

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 PMMar 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-on-call-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 • Flow run scheduled

Mar 22nd, 2023 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 • Block kubernetes job get client called

Mar 22nd, 2023 prefect.block.kubernetes-job.get_client.called

Resource

Block Document marvin-monday-slack-posts-k8s-job

Related Resources

1 resource

10:30:26 PM • Block kubernetes job get batch client called

Mar 22nd, 2023 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



PREFECT ASSOCIATE
CERTIFICATION

Can represent

- API calls
- state transitions
- changes in environment

Event Feeds



PREFECT ASSOCIATE
CERTIFICATION

Power several Cloud features

- flow run logs
- audit logs
- automations



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

- 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

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

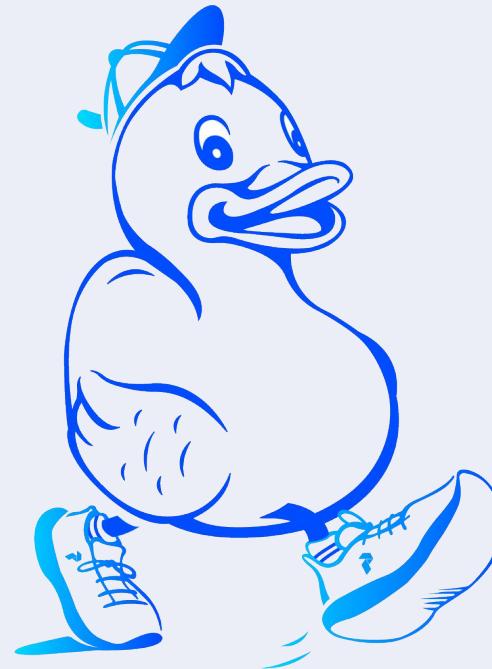


PREFECT ASSOCIATE
CERTIFICATION

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

```
@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

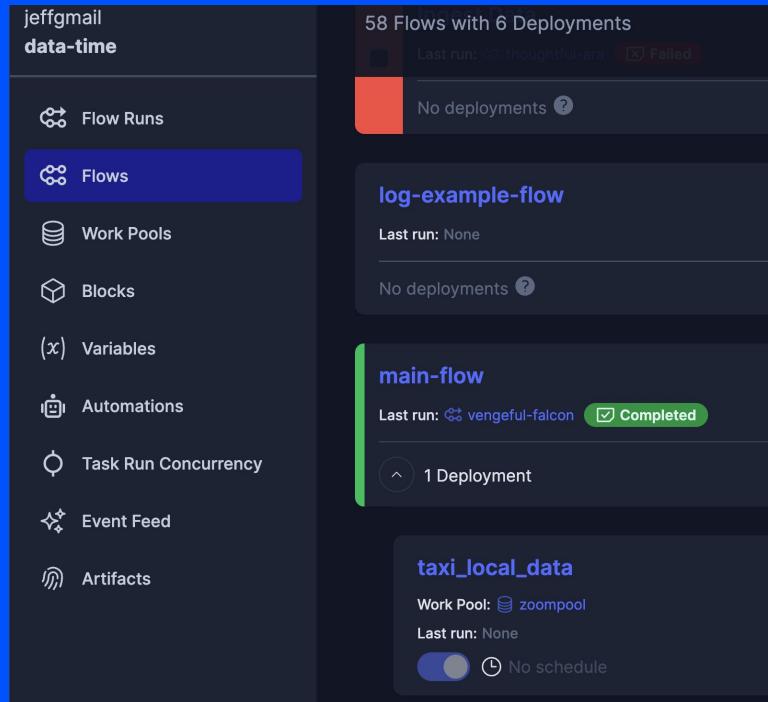


PREFECT ASSOCIATE
CERTIFICATION

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

Deployments in the UI

The deployment will live on the server



The screenshot shows the Prefect UI interface with several flow cards displayed:

- jeffgmai**
data-time
 - Flow Runs
 - Flows** (highlighted)
 - Work Pools
 - Blocks
 - Variables
 - Automations
 - Task Run Concurrency
 - Event Feed
 - Artifacts
- 58 Flows with 6 Deployments**
 - Last run: 🚀 thoughtful-ara (Failed)
 - No deployments ?
- log-example-flow**
 - Last run: None
 - No deployments ?
- main-flow**
 - Last run: 🚀 vengeful-falcon (Completed)
 - 1 Deployment
- taxi_local_data**
 - Work Pool: zoompool
 - Last run: None
 - No schedule



Prefect Projects

Prefect project



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.yaml

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

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

# preparation steps
build: null
push: null

# runtime steps
pull: null
```

Prefect Project - deployment.yaml

Describes base settings for a deployment produced from the project.

```
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/

- hidden directory where Prefect will store workflow metadata

.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



```
# 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/dsdcoder/ppcc-2023-05.git
    branch: main
    access_token: null
```

Prefect Project `prefect project init`



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



PREFECT ASSOCIATE
CERTIFICATION

- 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 / Create

01 Basic Information 02 Infrastructure Type 03 Configuration

Select the infrastructure you want to use to execute your flow runs

- Prefect Agent**
Execute flow runs on heterogenous infrastructure using infrastructure blocks.
- AWS Elastic Container Service** Beta
Execute flow runs within containers on AWS ECS. Works with existing ECS clusters and serverless execution via AWS Fargate. Requires an AWS account.
- Azure Container Instances** Beta
Execute flow runs within containers on Azure's Container Instances service. Requires an Azure account.
- Docker** Beta
Execute flow runs within Docker containers. Works well if you manage flow execution environments via Docker images. Requires access to a running Docker daemon.

Work Pools



PREFECT ASSOCIATE
CERTIFICATION



Google Cloud Run Beta

Execute flow runs within containers on Google Cloud Run. Requires a Google Cloud Platform account.



Kubernetes Beta

Execute flow runs within jobs scheduled on a Kubernetes cluster. Requires a Kubernetes cluster.



Local Subprocess Beta

Execute flow runs as subprocesses on a worker. Works well for local execution when first getting started.

Cancel

Previous

Next



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

- 1 Create a project in a new directory

```
$ mkdir my-project  
$ cd my-project  
$ prefect project init
```

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()
```

- 5 Deploy your flow

```
$ prefect deploy my-flow.py:say_hello -n 'my-deployment' -p my-pool
```

- 6 Start a run of the deployed flow from the CLI...

```
$ prefect deployment run Hello/my-deployment
```

Orchestration environment

Where Prefect Cloud or Server runs

- 3 Login to Prefect Cloud

```
$ prefect cloud login
```

- OR Start an open source Prefect Server

```
$ prefect server start
```



OR

from the UI

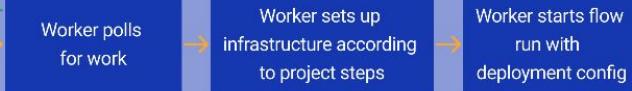


Execution environment

Where your flows run

- 4 Start a worker that polls your work pool

```
$ prefect worker start -p my-pool -t process
```



What just happened?



PREFECT ASSOCIATE
CERTIFICATION

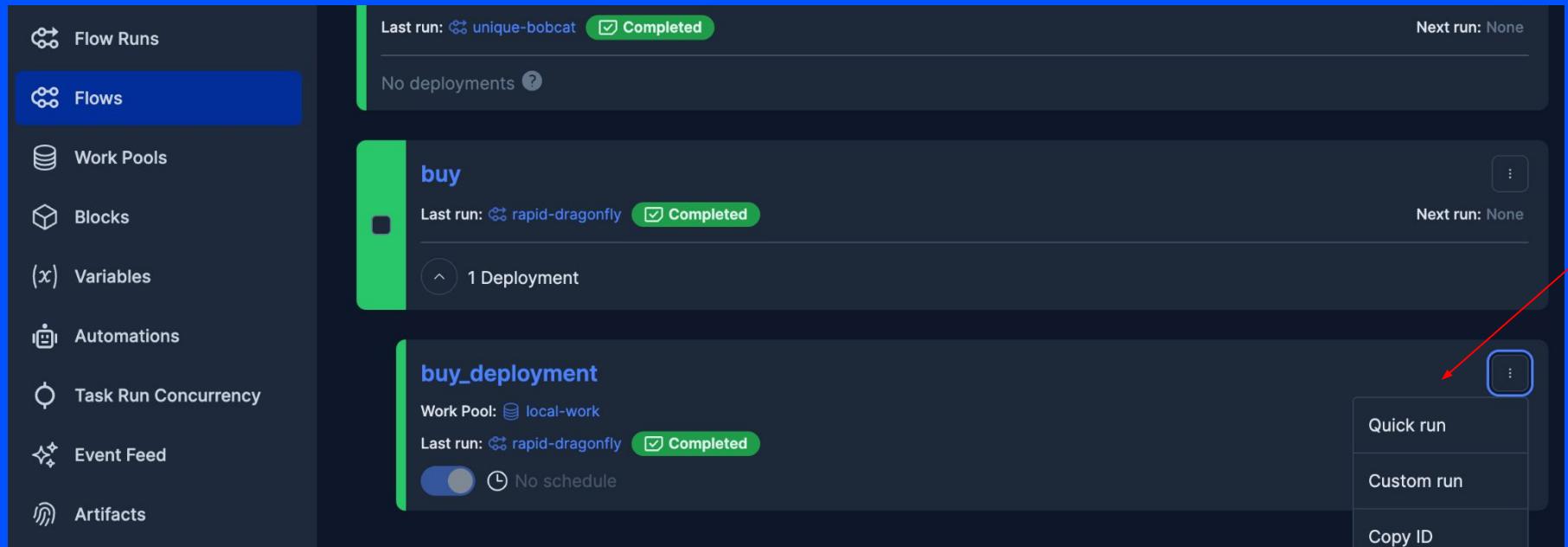
- Deployment *my-project-deploy* created on Prefect Cloud.
- Work pool *my_pool* created on Prefect Cloud.
- Worker started on local machine.
- Worker process polls Prefect Cloud, looking for scheduled work in the *my_pool* work pool.

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

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



The screenshot shows the Prefect UI interface. On the left, a sidebar menu lists various options: Flow Runs, Flows (which is selected and highlighted in blue), Work Pools, Blocks, Variables, Automations, Task Run Concurrency, Event Feed, and Artifacts. The main area displays two deployment cards. The first card is for a deployment named "buy", which has a green status bar indicating it was last run by "rapid-dragonfly" and completed successfully. Below this, a sub-card titled "1 Deployment" is shown. The second card is for a deployment named "buy_deployment", which also has a green status bar indicating it was last run by "rapid-dragonfly" and completed successfully. This card includes a toggle switch for "No schedule". To the right of these cards is a vertical ellipsis menu. A red arrow points to the top item in this menu, which is labeled "Quick run". Other items in the menu include "Custom run" and "Copy ID".

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
[INFO] 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
[INFO] 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
[INFO] 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

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

MODULE

105 - Deployments & Artifacts

105 Agenda

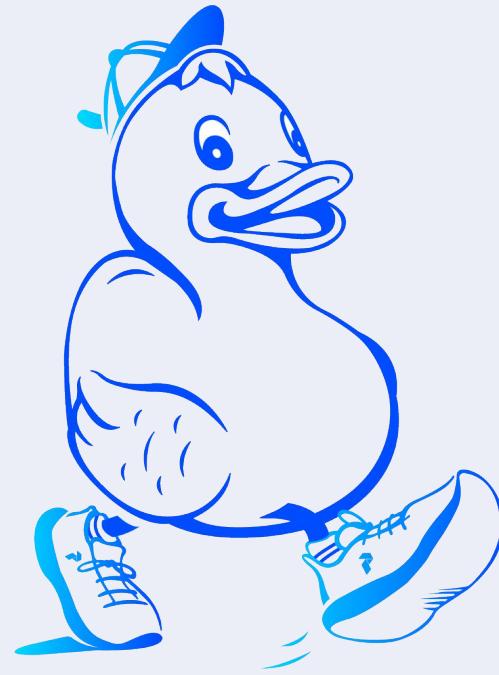
- Parameters
- Scheduling 
- Pausing and resuming schedules 
- Artifacts



PREFECT ASSOCIATE
CERTIFICATION

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

Deployments / taxi_s3_data

Runs **Parameters** Infra Overrides Description

2 parameters Search parameters

Key	Override	Default	Type
train_path		./data/green_tripdata_2021-01.parquet	string
val_path		./data/green_tripdata_2021-02.parquet	string

Flow
main-flow-s3

Work Pool
zoompool

Work Queue
default

Schedule
[Add](#)

Created
2023/05/12 07:34:35 PM

Created By
jeffgmail

Parameters



PREFECT ASSOCIATE
CERTIFICATION

Collaborators can run with custom values

Parameters

Default Custom JSON

train_path (Optional)

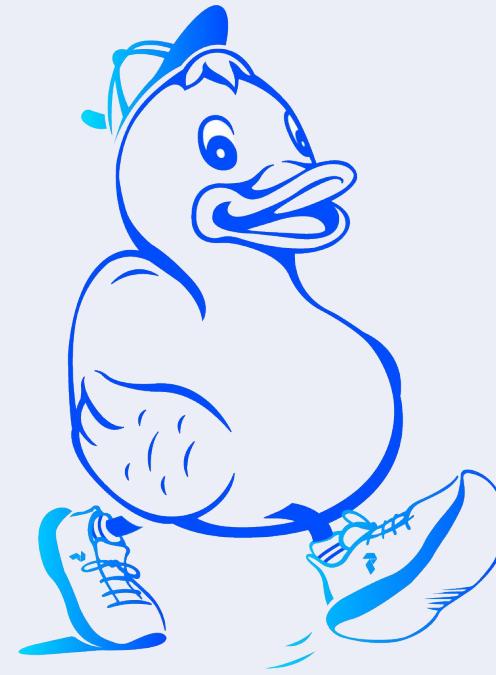
```
./data/green_tripdata_2021-01.parquet
```

val_path (Optional)

```
./data/green_tripdata_2021-02.parquet
```

Scheduling

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





PREFECT

Scheduling

Scheduling



PREFECT ASSOCIATE
CERTIFICATION

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

- Interval
- Cron
- RRule



Add a schedule in *deployment.yaml*



PREFECT ASSOCIATE
CERTIFICATION

Schedule created in
Cloud when you run
prefect deploy

```
description: null
entrypoint: null
flow_name: null
name: null
parameters: {}
schedule:
  cron: 0 0 * * *
  timezone: America/Chicago
tags: []
version: null
work_pool:
  job_variables: {}
  name: null
  work_queue_name: null
```

Add a schedule in the *prefect deploy* command

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

Scheduling from the UI



PREFECT ASSOCIATE
CERTIFICATION

Deployments / cat-fact-fetch

Runs Parameters Infra Overrides Description

0 Flow runs All run states Newest to oldest

No runs from the last 180 days

Flow
pipe

Work Pool
default-agent-pool

Work Queue
default

Infrastructure
anonymous-250496b3-c7db-419a-9f67-f5066980eba4

Schedule

Add



Scheduling from the UI



PREFECT ASSOCIATE
CERTIFICATION

Add schedule

Schedule type

Interval Cron RRule

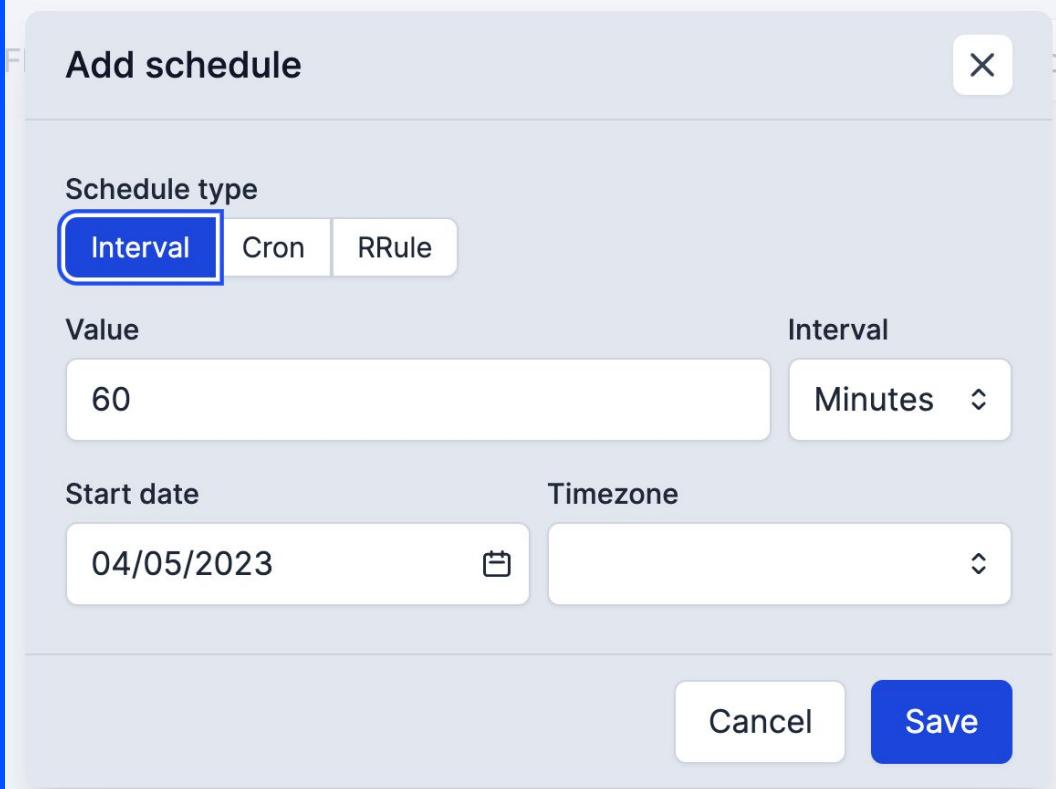
Value Interval

60 Minutes

Start date Timezone

04/05/2023

Cancel Save



Scheduling from CLI



```
prefect deployment set-schedule
```

```
weatherflow.py:fetch_weather --interval 40
```

Scheduling RRule



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

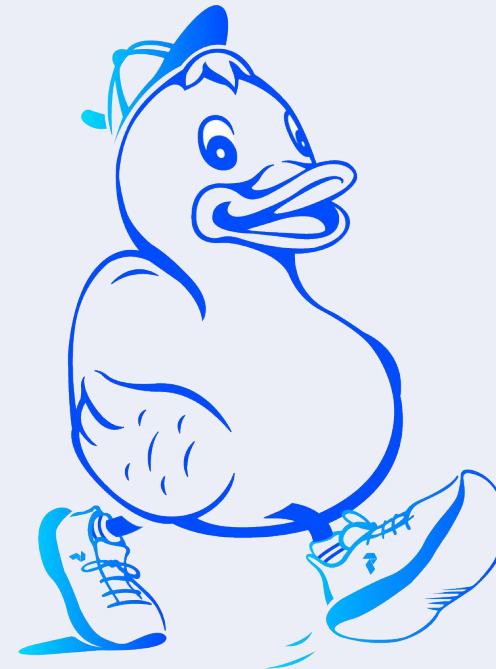
Or ask Marvin (another Prefect package) `pip install marvin`

```
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

A screenshot of the Prefect UI interface. At the top left is a blue header bar with a location pin icon and the text "Deployments". To its right is a dark grey header bar containing a small square icon, the text "fetch-weather/Python deploy file with params", the name "jeffgmai", and a blue toggle switch. A red arrow points from the text "Pause schedule from UI" to the blue toggle switch. On the far right of the dark grey bar is a vertical ellipsis (...).

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

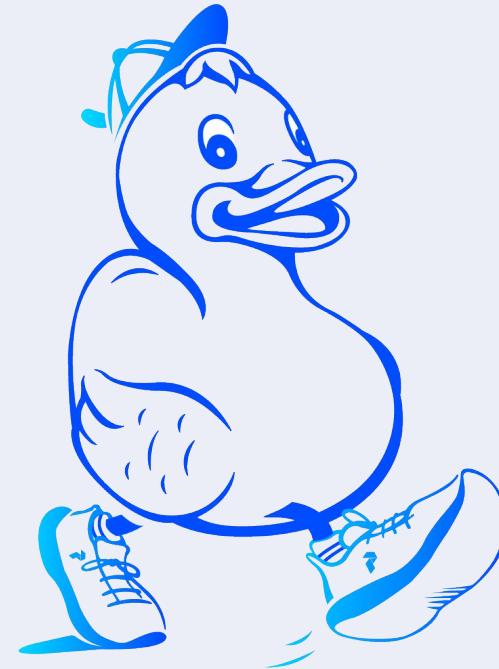
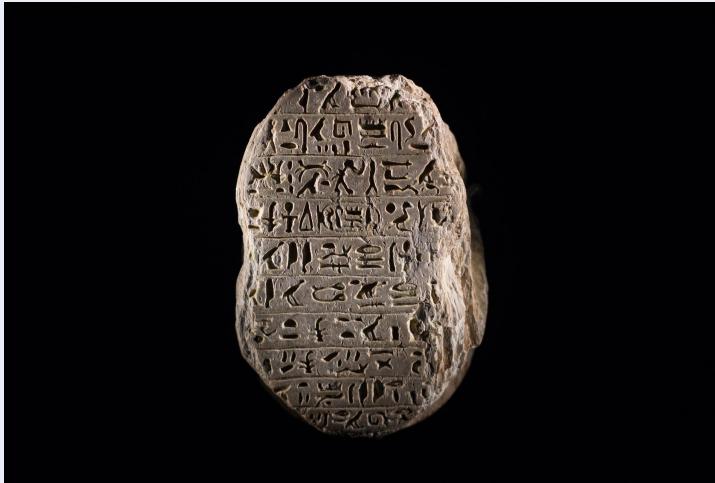
Artifacts



PREFECT
ASSOCIATE
CERTIFICATION

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

```
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

Access from Flow Runs or
Artifacts page



Very scientific weather report

Artifact Details Raw

Weather Report

Recent weather

Time	Revenue
Now	26.0
In 1 hour	28.0



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

- 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

MODULE

201 - Integrations & Docker

Integrations

Ducks integrate well.



Integrations



PREFECT ASSOCIATE
CERTIFICATION

docs.prefect.io/integrations/catalog/

Prefect Collections Catalog

Below you can find a list of all available Prefect Collections.

[prefect-airbyte](#)



[prefect-aws](#)



[prefect-azure](#)



[prefect-cubejs](#)



Maintained by Prefect

Maintained by Prefect

Maintained by Prefect

Maintained by Alessandro Lollo

[prefect-dask](#)



[prefect-databricks](#)



[prefect-dbt](#)



[prefect-email](#)



Maintained by Prefect

Maintained by Prefect

Maintained by Prefect

Maintained by Prefect

Integrations



PREFECT ASSOCIATE
CERTIFICATION

Python packages that add convenience

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

Integrations



PREFECT ASSOCIATE
CERTIFICATION

Prefect is Python-based and designed for flexibility

Use with most any Python library - no special
integration required 

201 Agenda

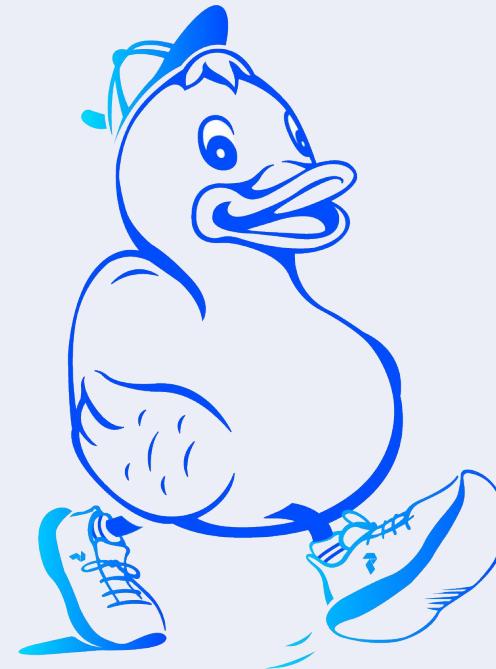
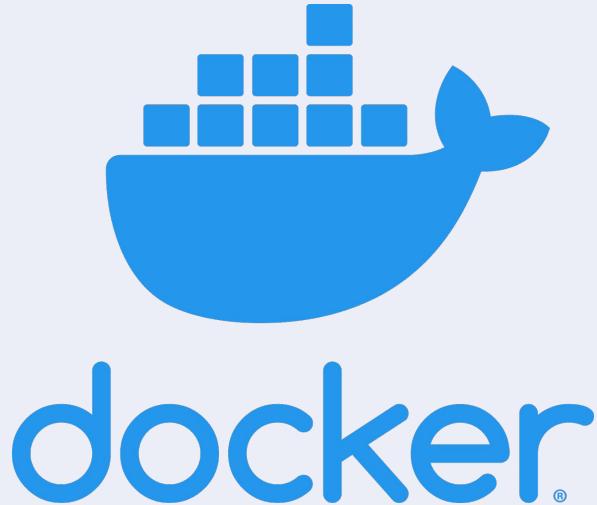


PREFECT ASSOCIATE
CERTIFICATION

Projects, Workers & Work Pools with Docker

Docker

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



Why use Docker?



PREFECT ASSOCIATE
CERTIFICATION

- 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

Prerequisites

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



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

Choose Docker

Work Pools / Create

Name

Description (Optional)

Flow Run Concurrency (Optional)

Unlimited

Type

The type of worker to run within this work pool. To learn more about workers, check out [the docs](#).

Prefect Agent

- Prefect Agent
- Process
- Ecs
- Azure Container Instance
- Docker
- Cloud Run
- Kubernetes

Work pools

Can specify a particular image your team created

The screenshot shows the 'Base Job Template' configuration page in Prefect, which is currently in Beta. The interface is dark-themed with light-colored text and highlights. At the top, there are two tabs: 'Defaults' (which is selected) and 'Advanced'. Below the tabs, a note states: 'The fields below control the default values for the base job template. These values can be overridden by deployments.' The configuration fields are organized into sections:

- Command (Optional)**: A text input field for the command to use when starting a flow run. It includes a note: 'The command to use when starting a flow run. In most cases, this should be left blank and the command will be automatically generated by the worker.'
- Environment Variables (Optional)**: A text input field for environment variables to set when starting a flow run. It contains the numbers 1, 2, and 3.
- Labels (Optional)**: A text input field for labels applied to infrastructure created by the worker using this job configuration. It contains the numbers 1, 2, and 3.
- Name (Optional)**: A text input field for the name given to infrastructure created by the worker using this job configuration.
- Image (Optional)**: A text input field for the image reference of a container image to use for created jobs. It includes a note: 'The image reference of a container image to use for created jobs. If not set, the latest Prefect image will be used.' The current value is 'docker.io/prefecthq/prefect:2-latest'.
- Image Pull Policy (Optional)**: A text input field for the image pull policy.

Work pools



PREFECT ASSOCIATE
CERTIFICATION

Base Job Template (Beta)

Defaults Advanced

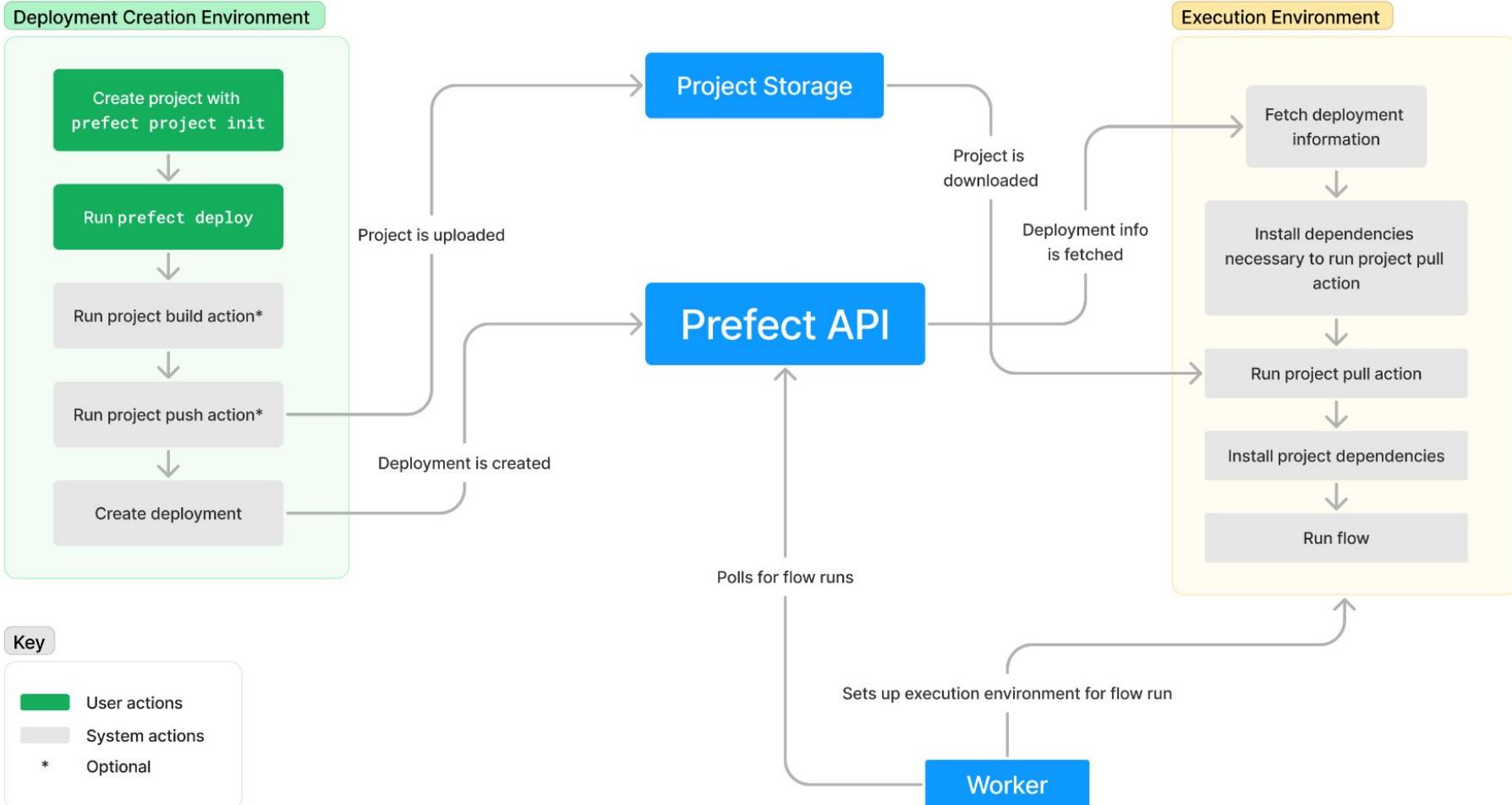
This is the JSON representation of the base job template. A work pool's job template controls infrastructure configuration for all flow runs in the work pool, and specifies the configuration that can be overridden by deployments.

ⓘ For more information on the structure of a work pool's base job template, check out [the docs](#).

```
{  
    "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





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

Prefect Project Docker Recipe



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

Prompts for image name and tag if not provided

Required inputs for 'docker' recipe

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

Prefect Project Docker Recipe



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

Docker - worker and work pool



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

```
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: discdive/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



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

```
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



Builds image

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

Flow run



PREFECT ASSOCIATE
CERTIFICATION

Docker configuration in the work pool can be overridden by
a deployment
(set in deployment.yaml)



Docker - deployment.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 deploy*
- Run deployment
- 🎉

Infrastructure - Docker Container



Check out Docker Desktop

A screenshot of the Docker Desktop application interface. At the top left, it says "Containers" and "Give feedback". Below that, a descriptive text states: "A container packages up code and its dependencies so the application runs quickly and reliably from one computing environment to another. [Learn more](#)". There is a toggle switch labeled "Only show running containers" which is turned off. To the right is a search bar with a magnifying glass icon and the placeholder "Search". Further right is a vertical ellipsis menu. The main area shows a table of containers. The columns are labeled: NAME, IMAGE, STATUS, PORT(S), STARTED, and ACTIONS. A header row has checkboxes for selecting multiple rows. The first row of data shows a checkbox, a hexagonal icon, the name "nano-tortoise", the image "prefecthq/r", the status "Exited", and a timestamp. The "ACTIONS" column contains icons for viewing logs, restarting, and deleting the container.

	NAME	IMAGE	STATUS	PORT(S)	STARTED	ACTIONS
<input type="checkbox"/>	nano-tortoise 26fed9f8e268	prefecthq/r	Exited			

201 Recap



PREFECT ASSOCIATE
CERTIFICATION

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



PREFECT

Bonus: Caching

Caching

What?

Why?

 task only

Requires persisting results



Caching: cache_key_fn

```
@task(cache_key_fn=task_input_hash)
```

```
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



```
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

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



Prefect Practitioner Certification Course



Congratulations!