PAL

Prefect Accelerated Learning

Slack

✅ Join Prefect Community Slack

✅ Join the *pal-* channel for the course

Norms

Norms

Code of conduct

- We expect all participants to be kind and respectful
- Reach out if you see or experience an issue

Goals

Goals

1. Competence with Prefect so you can build workflows you can trust
2. Connect with each other
3. Have fun! 🎉

Overview

Agenda for 1st half of the course

- Intro workflow orchestration and Prefect’s role
- 101: Prefect basics: Write workflows you can schedule and monitor
- 102: Orchestration and observation: Understand workflow state and guard against failure
- 103: Work with data and create automatic alerts

Agenda for 1st half of the course

- Intro workflow orchestration and Prefect’s role
- 101: Prefect basics: Write workflows you can schedule and monitor
- 102: Orchestration and observation: Understand workflow state and guard against failure
- 103: Work with data and create automatic alerts

Agenda for 2nd half of the course

- 104: Easily switch infrastructure and manage teams with work pool-based deployments
- 105: Workflow design patterns: Compose workflows that meet your needs
- 106: Interact with workflows, run tasks concurrently, create advanced automation triggers

Intro to Orchestration

What is orchestration?

- Automating, scheduling, and managing the flow of data across various pipelines and systems.
- Ensures that each step in a workflow runs in the correct order, under the right conditions, and at the right time.
- Failure recovery when needed.

Orchestration benefits across disciplines

Data engineers

- Reduce pipeline errors
- Increase productivity through automation
- At-a-glance understanding

Data science & ML engineers

- Iterate on ML models faster
- Reduce data processing time
- Move to production quickly

AI engineers

- Manage agentic workflow troubleshooting
- Unify orchestration across multiple LLM tasks

Data platform engineers

- Self-serve, turn-key infrastructure setup
- Faster onboarding
- Compute governance

What is Prefect?

Prefect is an orchestration and observability platform that empowers developers to build resilient workflows you can trust. 💙

Prefect data workflow orchestration

Automation

Flexible Orchestration in Pure Python

Resilience

Faster Recovery with Resilient Code

Scalability

Scalable compute resources and governance

Outcomes

- Save time ⏱️
- Save money 💰
- Increase productivity 🚀

101 Prefect basics: Create a workflow you can schedule and observe

101 Agenda

- Setup
- From Python function to Prefect flow
- Create a deployment with *.serve()*
- Run a deployment
- Schedule a deployment
- Parameters
- Helpful resources

Install most recent version of Prefect 3

pip install -U prefect

Or, if using uv

uv pip install -U prefect

⭐ You can do this and any of the other items you'll see on upcoming slides during the first lab. ⭐

Prefect information in the CLI

prefect version

```
Version:              3.2.7
API version:          0.8.4
Python version:       3.12.4
Git commit:           d4d9001e
Built:                Fri, Feb 21, 2025 7:39 PM
OS/Arch:              darwin/arm64
Profile:              prefect-cloud
Server type:          cloud
Pydantic version:     2.10.6
Integrations:
  prefect-dask:       0.3.2
```

Prefect has two options for server interaction

1. Self-host a Prefect server
   a. You spin up a local server
   b. Backed by SQLite db (or PostgreSQL)
2. Use the Prefect Cloud platform
   a. Free tier
   b. Organization management capabilities on other tiers
   a. Additional features such as event webhooks, push work pools, managed work pools
   c. No database management required

To the Cloud

Prefect Cloud

Go to app.prefect.cloud in browser

- Sign up or sign in
- Use a free personal account if you don’t want to use an organization account

Prefect Cloud

Authenticate your CLI

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
```

Select ***Log in with a web browser***

Creates and saves an API key for you 🔑

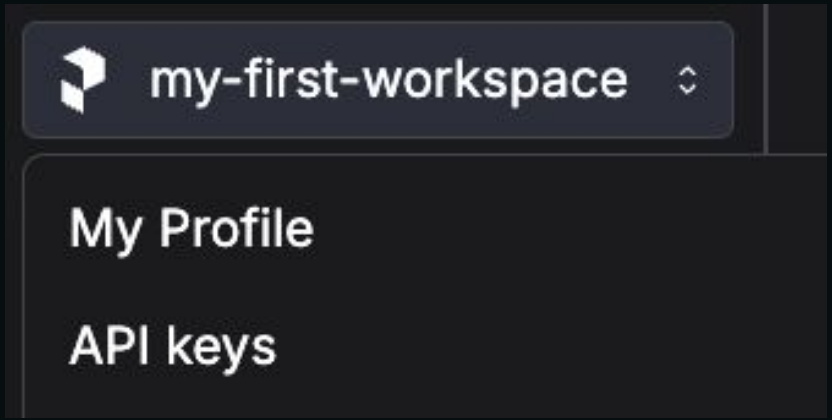
Prefect Cloud

OR, if UI doesn’t work:

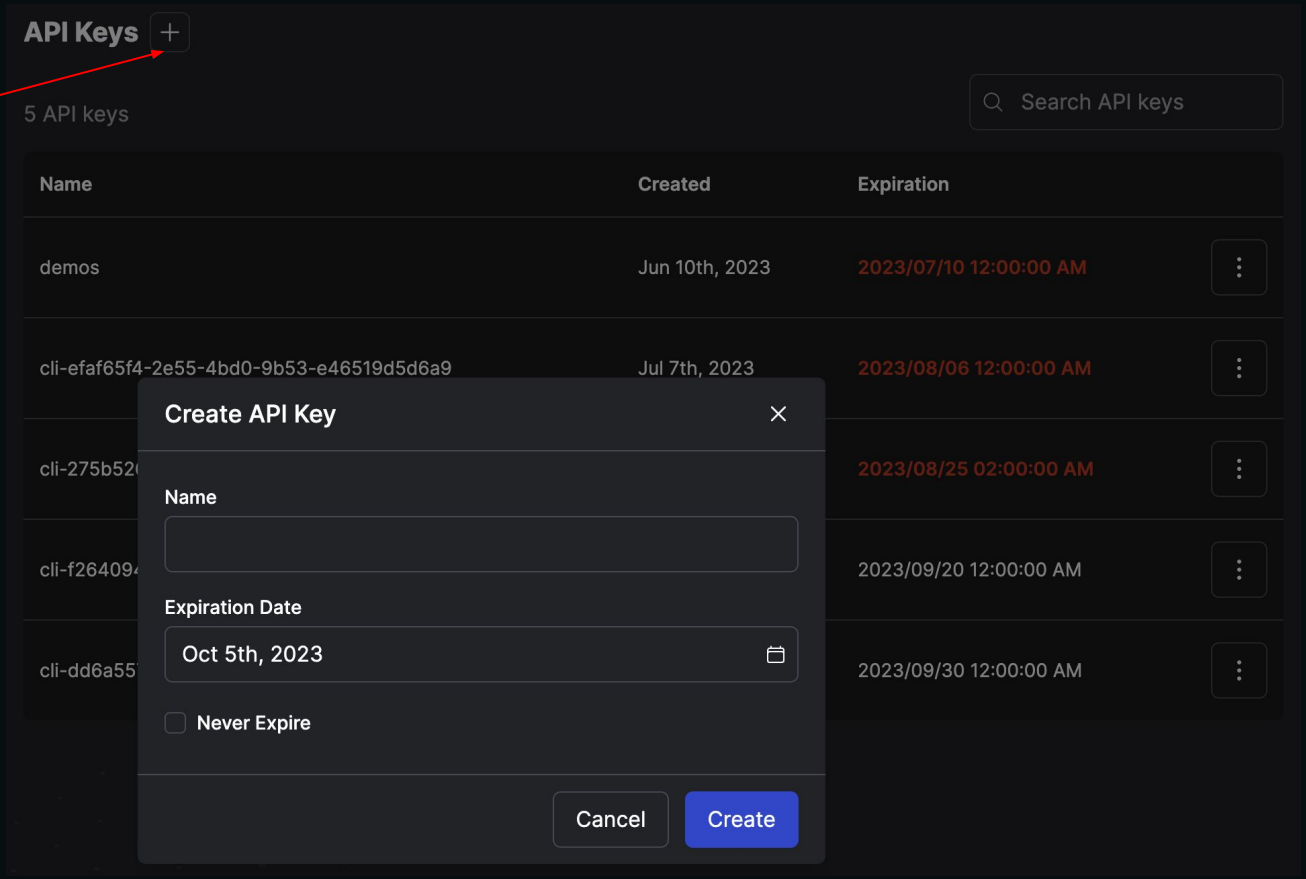
- Select *Paste an API key*
- Manually create an API key from Prefect Cloud in the UI

Prefect Cloud - API key

(Top left in UI)



Prefect Cloud - API key



Prefect Cloud - API key

Paste API key at terminal prompt

```
> Paste an API key
Paste your API key:
```

To just switch between workspaces from the CLI

prefect cloud workspace set

```
? Which account would you like to use? [Use arrows to move; enter to select]
> prefect-sandbox
  sales-engineering
  prefect-technologies
  jeffprefectio
```

Prefect profiles

Prefect profiles

- Persistent settings for interacting with Prefect
- One profile active at all times
- Common to switch between:
  - Cloud and a self-hosted Prefect server
  - Cloud workspaces
  - Saved settings such as logging level

Prefect profiles

List: *prefect profile ls*

```
Available Profiles:

* default
  local
  jeffmshale
  gh2
  prefect-more
```

Prefect profiles

Profiles live in ***~/.prefect/profiles.toml*** 📁

```
active = "sandbox-jeff"

[profiles.default]
PREFECT_API_URL = "http://127.0.0.1:4200/api"
PREFECT_DEFAULT_WORK_POOL_NAME = "default-pool"
PREFECT_LOGGING_LEVEL = "DEBUG"

[profiles.qawork]
PREFECT_API_KEY = "pnu_GSLLSpUFz83ZgecfLsEeBy9TDdAWqu3xAQX"
PREFECT_API_URL = "https://api.stg.prefect.dev/api/accounts/8e8e0fcc-53a5-46f4-80b1-d8fdf4fae7
PREFECT_LOGGING_LEVEL = "DEBUG"
PREFECT_DEFAULT_DOCKER_BUILD_NAMESPACE = "us-central1-docker.pkg.dev/prefect-sbx-community-eng

[profiles.storage-demo]
PREFECT_API_KEY = "pnu_F16ZsLxoen5gjlQHGkDCatS7LiUBO42xgfQ"
PREFECT_API_URL = "https://api.prefect.cloud/api/accounts/9b649228-0419-40e1-9e0d-44954b5c0ab6
```

Prefect profiles

Profile stays active until you switch to another profile

Contains:

1. API URL
2. API key for Prefect Cloud (if using Cloud)
3. Optional configuration

Prefect profiles

Create: *prefect profile create my_cloud_profile*

Inspect: *prefect profile inspect my_cloud_profile*

Switch: *prefect profile use my_cloud_profile*

Flows: Add superpowers to your Python 🦸

Project

We are consulting for a company that uses weather forecast data in financial products.

We'll fetch and use weather forecast data from Open-Meteo 🌦️ 🌡️

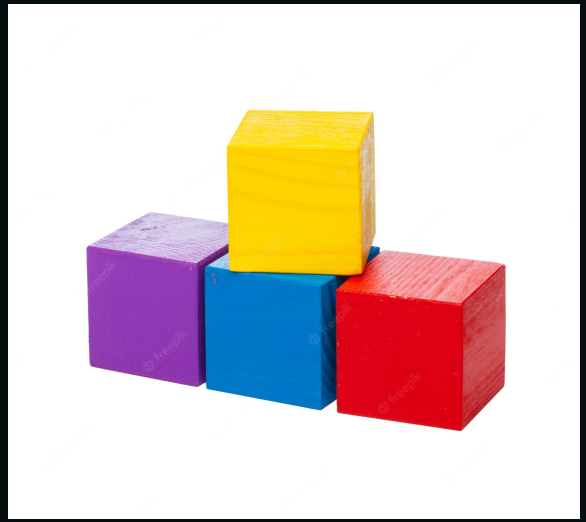
open-meteo.com

Start: basic Python function

```python
import httpx


def fetch_weather(lat: float = 38.9, lon: float = -77.0):
    base_url = "https://api.open-meteo.com/v1/forecast/"
    temps = httpx.get(
        base_url,
        params=dict(latitude=lat, longitude=lon, hourly="temperature_2m"),
    )
    forecasted_temp = float(temps.json()["hourly"]["temperature_2m"][0])
    print(f"Forecasted temp C: {forecasted_temp} degrees")
    return forecasted_temp


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

Flows

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



Make it a flow

```python
import httpx
from prefect import flow


@flow()
def fetch_weather(lat: float = 38.9, lon: float = -77.0):
    base_url = "https://api.open-meteo.com/v1/forecast/"
    temps = httpx.get(
        base_url,
        params=dict(latitude=lat, longitude=lon, hourly="temperature_2m"),
    )
    forecasted_temp = float(temps.json()["hourly"]["temperature_2m"][0])
    print(f"Forecasted temp C: {forecasted_temp} degrees")
    return forecasted_temp


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

Run the code: *python my_file.py*

```
08:13:56.632 | INFO    | prefect.engine - Created flow run 'qualified-lorikeet' for flow 'fetch-weather'
08:13:56.633 | INFO    | prefect.engine - View at https://app.prefect.cloud/account/9b649228-0419-40e1-9e0d-44
954b5c0ab6/workspace/d137367a-5055-44ff-b91c-6f7366c9e4c4/runs/flow-run/0219f38e-a1b5-437e-b271-749c22a1e929
Forecasted temp C: 10.4 degrees
08:13:57.381 | INFO    | Flow run 'qualified-lorikeet' - Finished in state Completed()
```

Check it out your flow run from the **Runs** page in the UI

Runs / **fetch-weather** / **qualified-lorikeet** Completed

This flow run did not generate any task or subflow runs

Search | Display

qualified-lorikeet +3 08:13:56 AM
- prefect.flow-run.Pending 08:13:56 AM
- prefect.flow-run.Running 08:13:56 AM
- prefect.flow-run.Completed 08:13:57 AM

qualified-lorikeet Flow run

Properties

State	Completed
Scheduled start	2024/10/16 08:13:56 AM
Start	2024/10/16 08:13:56 AM
End	2024/10/16 08:13:57 AM
Duration	1s
Run count	1 / 1
Creator	jeffprefectio
Flow	fetch-weather
Run	qualified-lorikeet
Deployment	None
Work pool	None
Work queue	None
Automation	None

Tags

Parameters

```
{
  "lat": 38.9,
  "lon": -77
}
```

Flows give you

- Auto logging
- State tracking info sent to API
- Input arguments type checked/coerced
- Timeouts can be enforced
- Lots of other benefits you'll see soon 🚀

Deployments

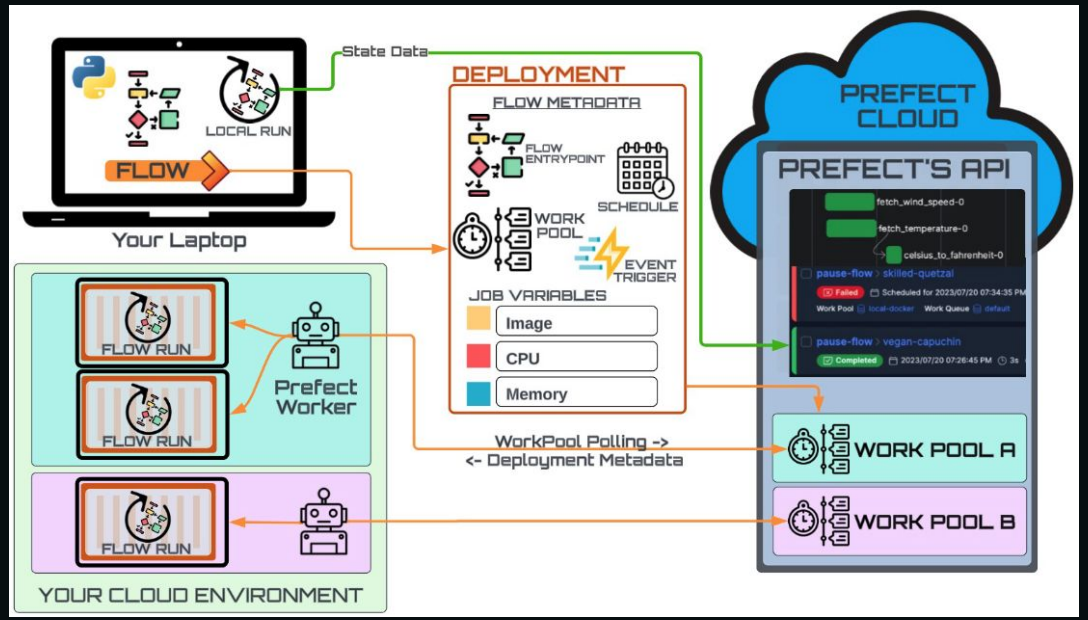
Deployments

Turn your Python workflow into an interactive application! 🎉

- Switch infrastructure easily
- You and teammates can run:
  - manually (from the UI or CLI)
  - on a schedule
  - in response to an automation trigger

Deployments

- Server-side representation of a flow
- Contains metadata for remote orchestration



.serve() method

Create a deployment by calling flow function's *.serve()* method

```
if __name__ == "__main__":
    fetch_weather.serve(name="deploy-1")
```

.serve() method

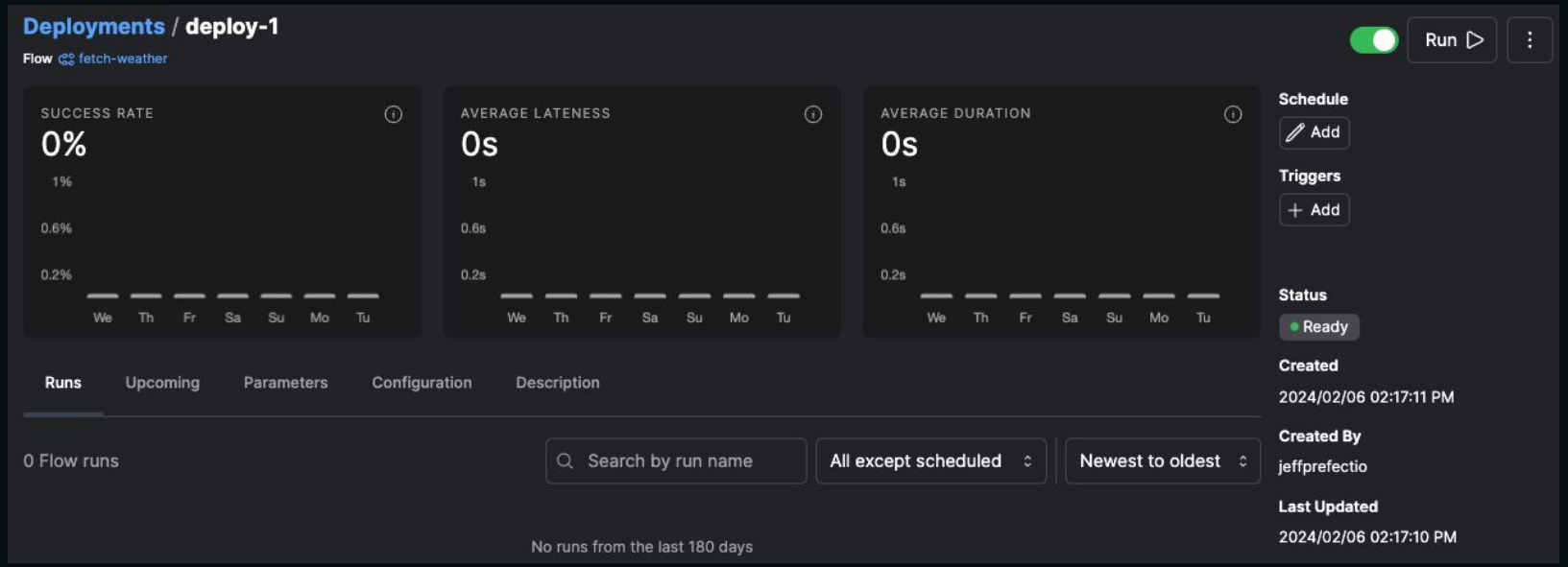
Run the script - creates a deployment and starts a process that polls for scheduled runs

```
Your flow 'fetch-weather' is being served and polling for scheduled runs!

To trigger a run for this flow, use the following command:

        $ prefect deployment run 'fetch-weather/deploy-1'

You can also run your flow via the Prefect UI:
https://app.prefect.cloud/account/55c7f5e5-2da9-426c-8123-2948d5e5d94b/workspace/7ad1ef2f-2f9c-49b5-b
29f-4e0b3760d4c6/deployments/deployment/73c53509-8e7f-4924-a208-9d9bf2a50558
```

You just made a deployment!

See the deployment in the UI

Deployment page



Run a deployment

Run manually from UI: **Run -> Quick run**

Deployments / **deploy-1**

Flow fetch-weather

Run

Quick run

Custom run

SUCCESS RATE

100%

100% 60% 20%

Tu Th Sa Mo

AVERAGE LATENESS

0s

1s 0.6s 0.2s

We Th Fr Sa Su Mo Tu

AVERAGE DURATION

1s

1s 0.6s 0.2s

Tu We Th Fr Sa Su Mo Tu

Runs | Upcoming | Parameters | Configuration | Description

2 Flow runs

Search by run name

All except scheduled

Newest to oldest

fetch-weather > congenial-parrot

Completed 2024/05/21 08:03:41 AM (72d 21h late) 1s None

Deployment deploy-1

fetch-weather > complex-pogona

Completed 2024/02/06 02:23:05 PM 2s None

Deployment deploy-1

Schedules

+ Schedule

Triggers

+ Add

Status

Ready

Created

2024/02/06 02:17:11 PM

Created By

jeffprefectio

Last Updated

2024/05/21 08:03:36 AM

Updated By

jeffprefectio

Entrypoint

weather1-serve.py:fetch_weather

Path

.

Flow ID

f5ccfd6c-46ae-4388-b5ef-2f46e3f88798

Deployment ID

d55ce219-510b-421f-99da-1934dab91995

Deployment Version

View the flow run logs in the UI (or CLI)

Runs / fetch-weather / deploy-1 / inescapable-griffin Completed

Retry

8:19:41 AM 8:19:42 AM 8:19:43 AM 8:19:44 AM 8:19:45 AM 8:19:46 AM 8:19:47 AM 8:19:48 AM 8:19:49 AM 8:19:50 AM 8:19:51 AM 8:19:52 AM 8:19:53 AM 8:19:54 AM 8:19:55 AM 8:19:56 AM

This flow run did not generate any task or subflow runs

Search Display

inescapable-griffin +8	08:19:50 AM
prefect.flow-run.Scheduled	08:19:41 AM
Runner 'deploy-1' submitting flow run 'df6e6327-6c53-463a-b49f-db748a9325ae'	08:19:48 AM
prefect.flow-run.Pending	08:19:48 AM
Opening process...	08:19:48 AM
Completed submission of flow run 'df6e6327-6c53-463a-b49f-db748a9325ae'	08:19:48 AM
prefect.flow-run.Running	08:19:50 AM
prefect.flow-run.Completed	08:19:50 AM
Process for flow run 'inescapable-griffin' exited cleanly.	08:19:51 AM

inescapable-griffin Flow run

Properties

State	Scheduled
Scheduled start	2024/10/16 08:19:41 AM
Start	None
Duration	0s
Run count	0 / 1
Creator	jeffprefectio
Flow	fetch-weather
Run	inescapable-griffin
Deployment	deploy-1
Work pool	None
Work queue	None
Automation	None

Tags

Parameters

Configuration

Logs

```
Runner 'deploy-1' submitting flow run 'df6e6327-6c53-463a-b49f-db748a9325ae'   08:19:48 AM INFO
Opening process...                                                             08:19:48 AM INFO
Completed submission of flow run 'df6e6327-6c53-463a-b49f-db748a9325ae'        08:19:48 AM INFO
Process for flow run 'inescapable-griffin' exited cleanly.                     08:19:51 AM INFO
```

Run deployment manually from CLI

prefect deployment run my_entrypoint_flow:my_deployment

.serve()

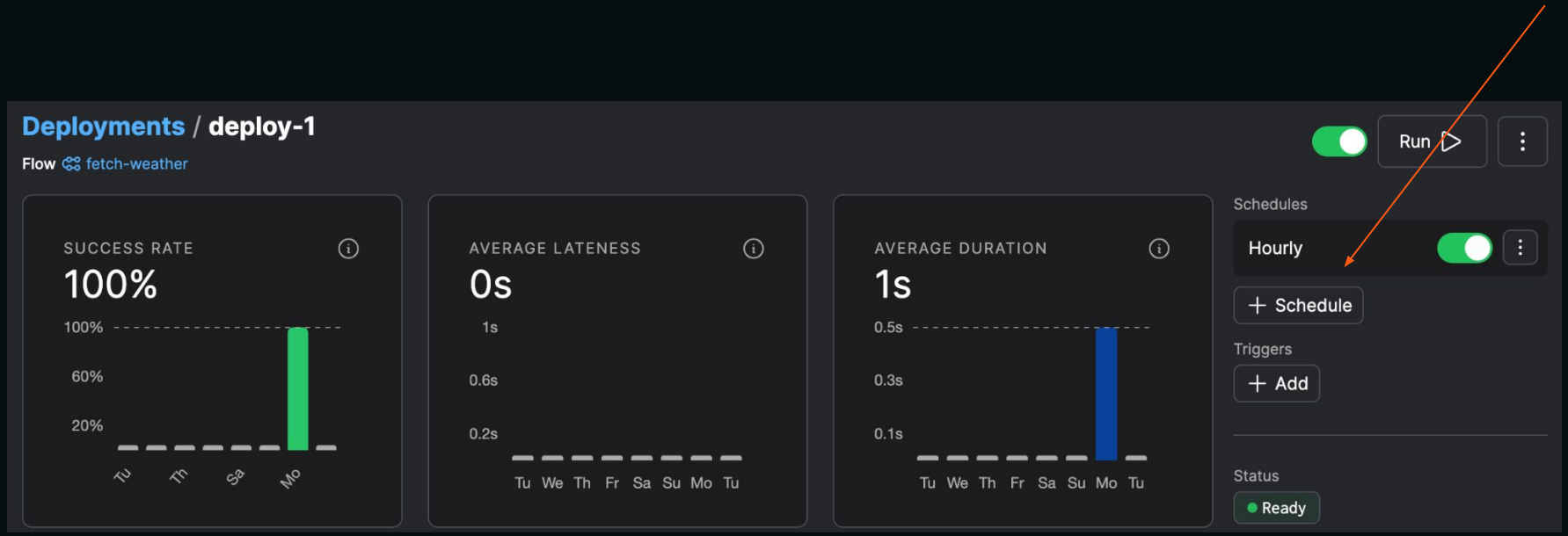
Shut down the process with ***control* + *c***

Scheduling

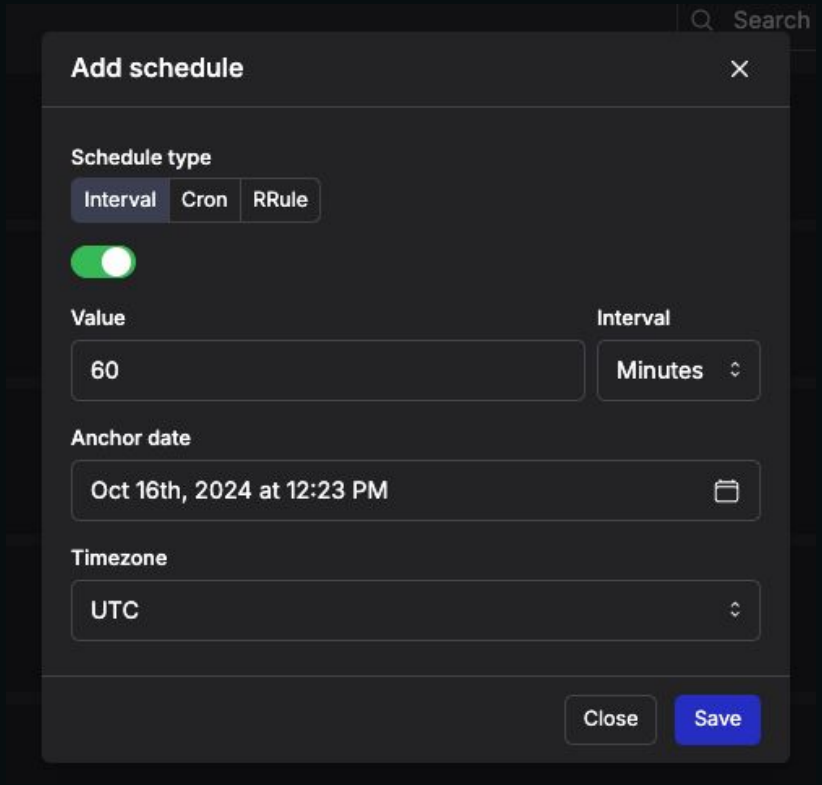
Create a deployment schedule

1. When create deployment
2. After deployment creation in UI or CLI

Create, pause, and delete schedules from UI



Create schedule from UI



Add schedule when create deployment with .*serve()*

```python
import httpx
from prefect import flow


@flow()
def fetch_weather(lat: float = 38.9, lon: float = -77.0):
    base_url = "https://api.open-meteo.com/v1/forecast/"
    temps = httpx.get(
        base_url,
        params=dict(latitude=lat, longitude=lon, hourly="temperature_2m"),
    )
    forecasted_temp = float(temps.json()["hourly"]["temperature_2m"][0])
    print(f"Forecasted temp C: {forecasted_temp} degrees")
    return forecasted_temp


if __name__ == "__main__":
    fetch_weather.serve(name="deploy-scheduled", cron="* * * * *")
```

Schedule types

- Interval
- Cron
- RRule

RRule

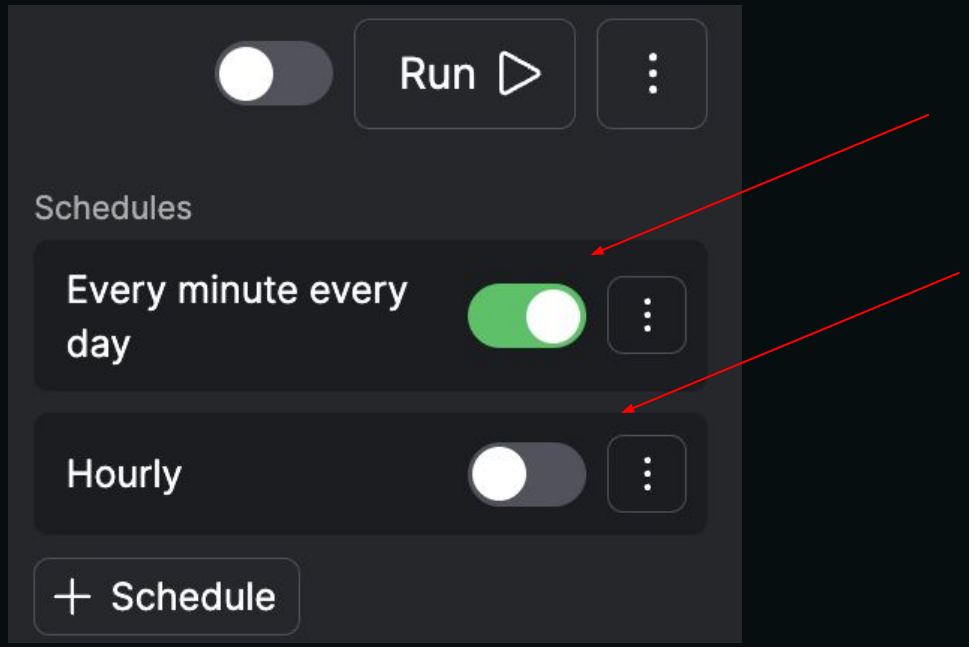
RRule cheat sheet: https://jkbrzt.github.io/rrule/

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 resuming deployment schedules

Pause/resume deployment schedules from UI



Note ⏸️

Shutting down your *serve* process pauses a deployment's schedules

Parameters

Parameters - argument values for entrypoint flow function

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

Parameter options

1. Make default arguments in flow function definition
2. Can override at deployment creation
3. Can override both of the above at runtime

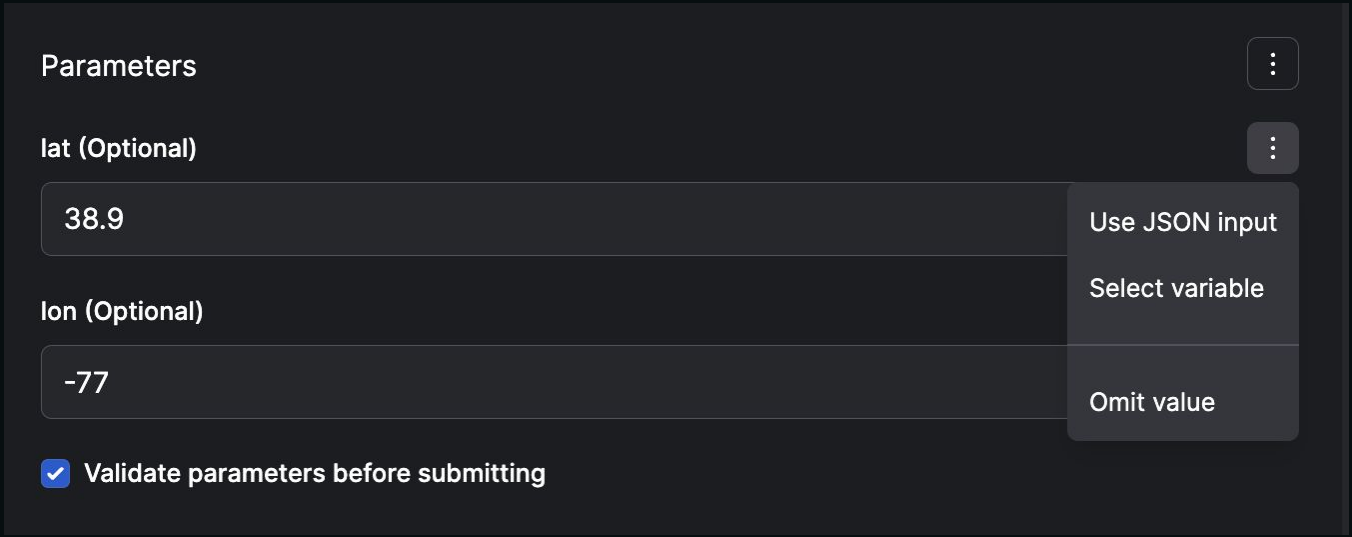
Parameters at deployment creation time

Can specify in *.serve()*

```
if __name__ == "__main__":
    fetch_weather.serve(name="deploy-params", parameters={"lat": 11, "lon": 12})
```

Parameters in the UI at runtime

Collaborators can run with custom values in a **Custom run** in the UI



Parameters from the CLI at runtime

prefect deployment run parameterized/dev --param user=Marvin --param answer=42

OR

prefect deployment run parameterized/dev --params '{"user": "Marvin", "answer": 42}'

Resources

Docs

Use the docs

docs.prefect.io

Prefect codebase

github.com/PrefectHQ/prefect

- Dig into the code
- Create an issue
- Make a PR
- Give it a ⭐

101 Recap

You've seen how to get started with Prefect!

- *prefect version, login*, profiles
- From Python function to Prefect flow
- Create a deployment with *flow.serve()*
- Run a deployment
- Create and pause schedules
- Resources: docs, Prefect GitHub repo

Recap key terms

Flow = a workflow

Flow run = an individual run of a flow

Deployment = a flow + orchestration capabilities

- Can schedule
- Can run remotely
- Other team members can access

Lab 101

Course GitHub Repository

Updated GitHub link in video description!

101 Lab

Use Open-Meteo API to fetch your first weather forecast:

- Authenticate your CLI to Prefect Cloud
- Fine to use a personal account or an organization test workspace
- Take a function that fetches data and make it a flow
- Use *.serve()* method to deploy your flow
- Run your flow from the UI
- Create a schedule for your deployment
- Shut down your *serve* process and restart it

101 Lab

- Stretch 1: Run a deployment from the CLI, override the params
- API docs: open-meteo.com/en/docs
- Example: wind speed for the last hour:

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

- If you are on a personal plan and bump up against the deployment limit cap, delete unneeded deployments