105 - Workflow design patterns: Compose workflows that meet your needs

PREFECT

Quick Recap

How Prefect works - Architecture Overview

Your Laptop

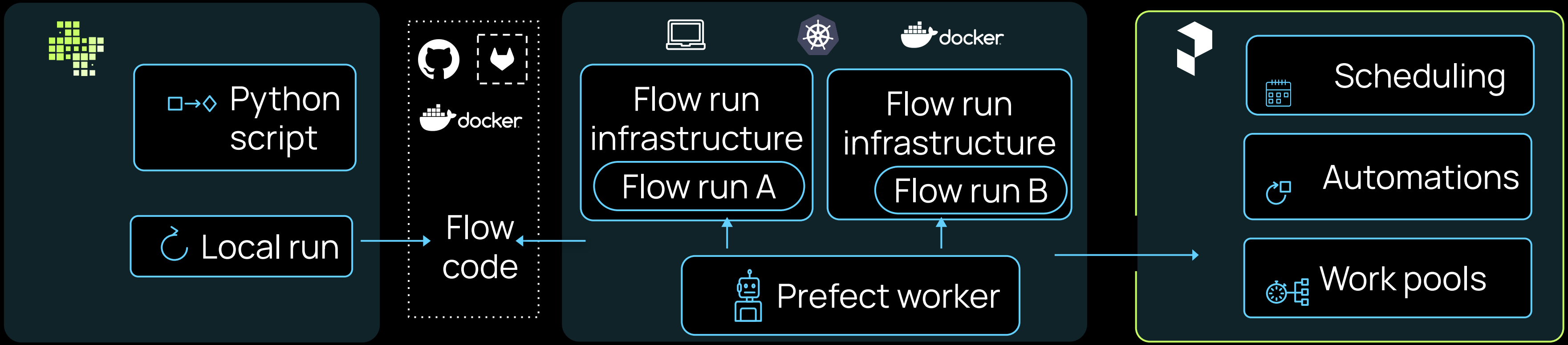
Write data workflow logic through Prefect's Python SDK

Your Execution Environment

Run data workflows on any infrastructure environment

Prefect Cloud

Understand & manage with the Prefect dashboard



105 Agenda

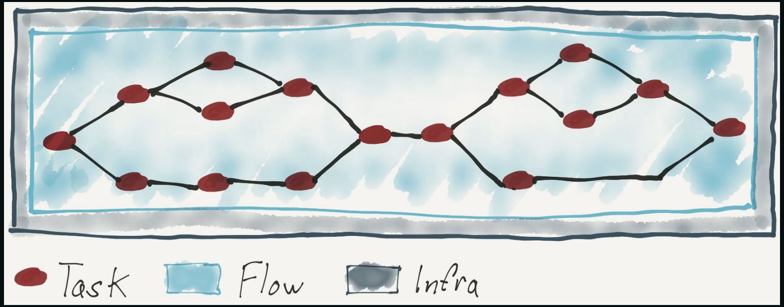
Workflow pattern archetypes

- Monoflow
- Subflows
- *run_deployment*
- Event-driven
  - Deployment triggers
  - Custom events
  - Webhooks

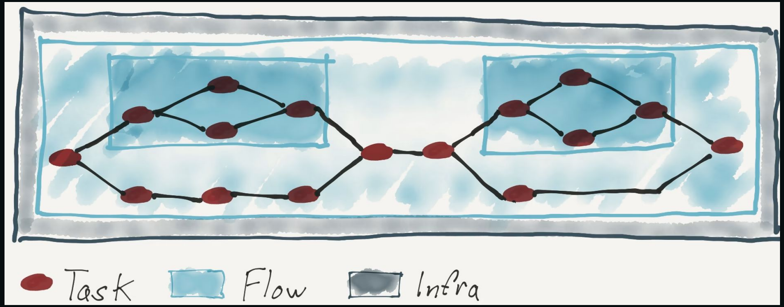
- Tasks alone

Workflow patterns

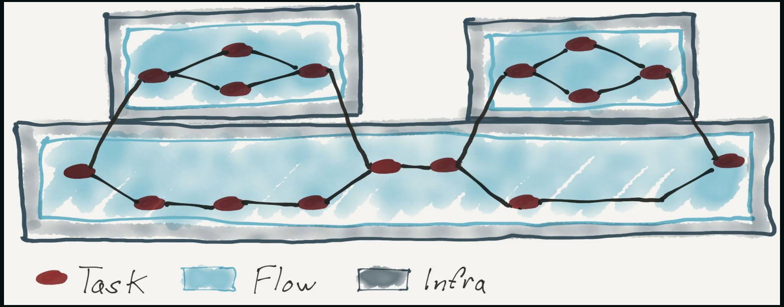
Workflow patterns - based on prefect.io/blog/workflow-design-patterns



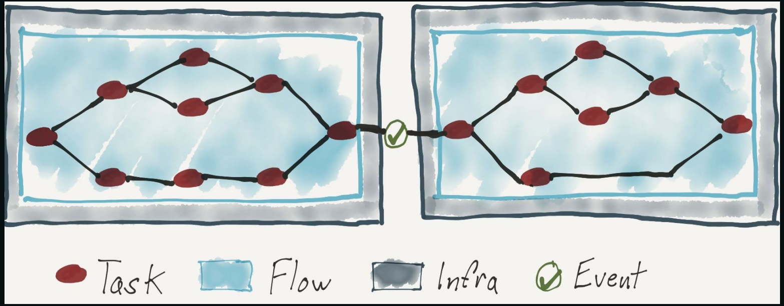
Flow of tasks



Flow of nested flows

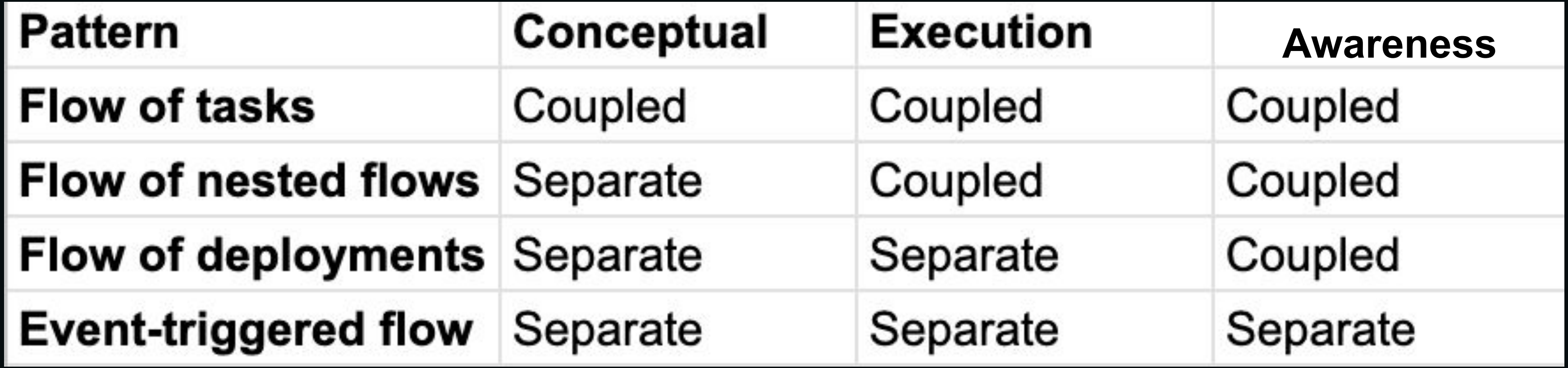


Flow of deployments



Event-triggered flow

When to use which?



Pattern	Conceptual	Execution	Awareness
Flow of tasks	Coupled	Coupled	Coupled
Flow of nested flows	Separate	Coupled	Coupled
Flow of deployments	Separate	Separate	Coupled
Event-triggered flow	Separate	Separate	Separate

UI/ Organization

Infra/ Process

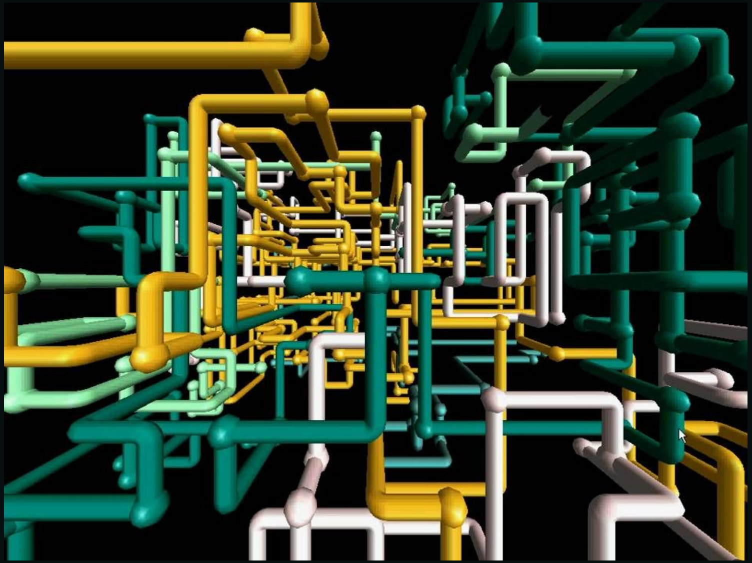
Context/ State

Nested flows

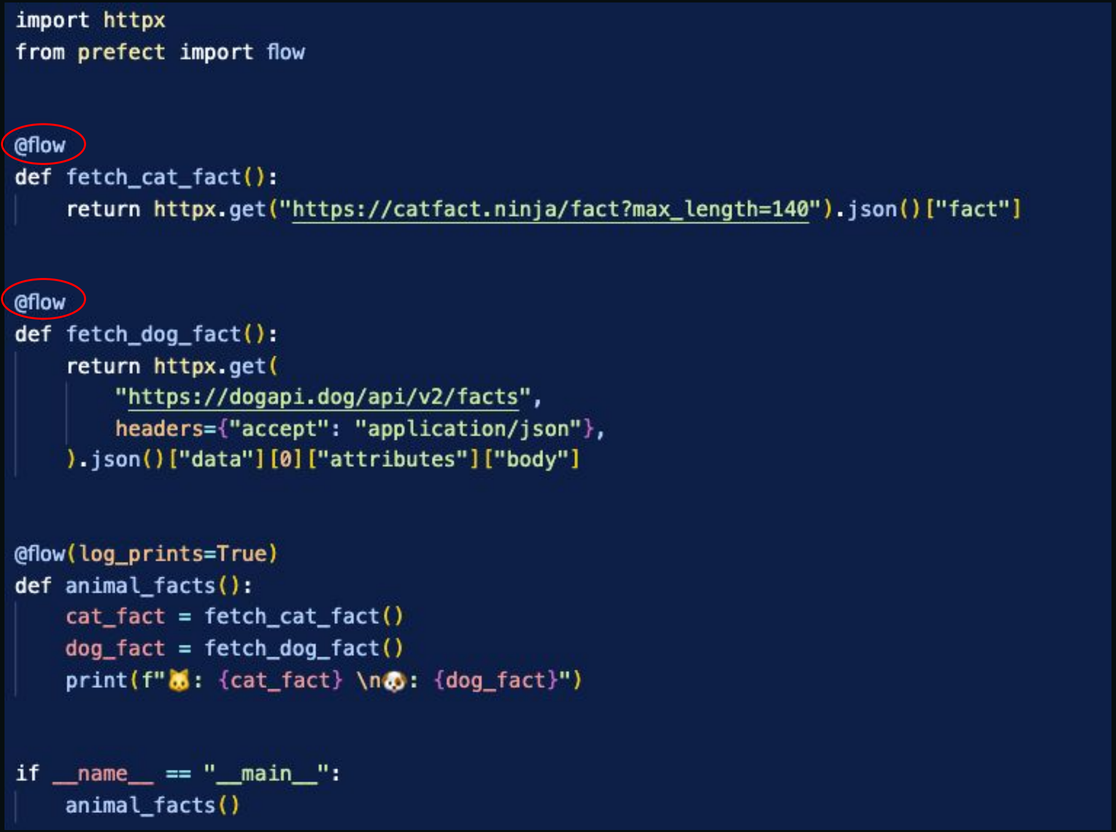
(AKA subflows)

Nested flows

- A flow called from another flow
- Useful for grouping related tasks

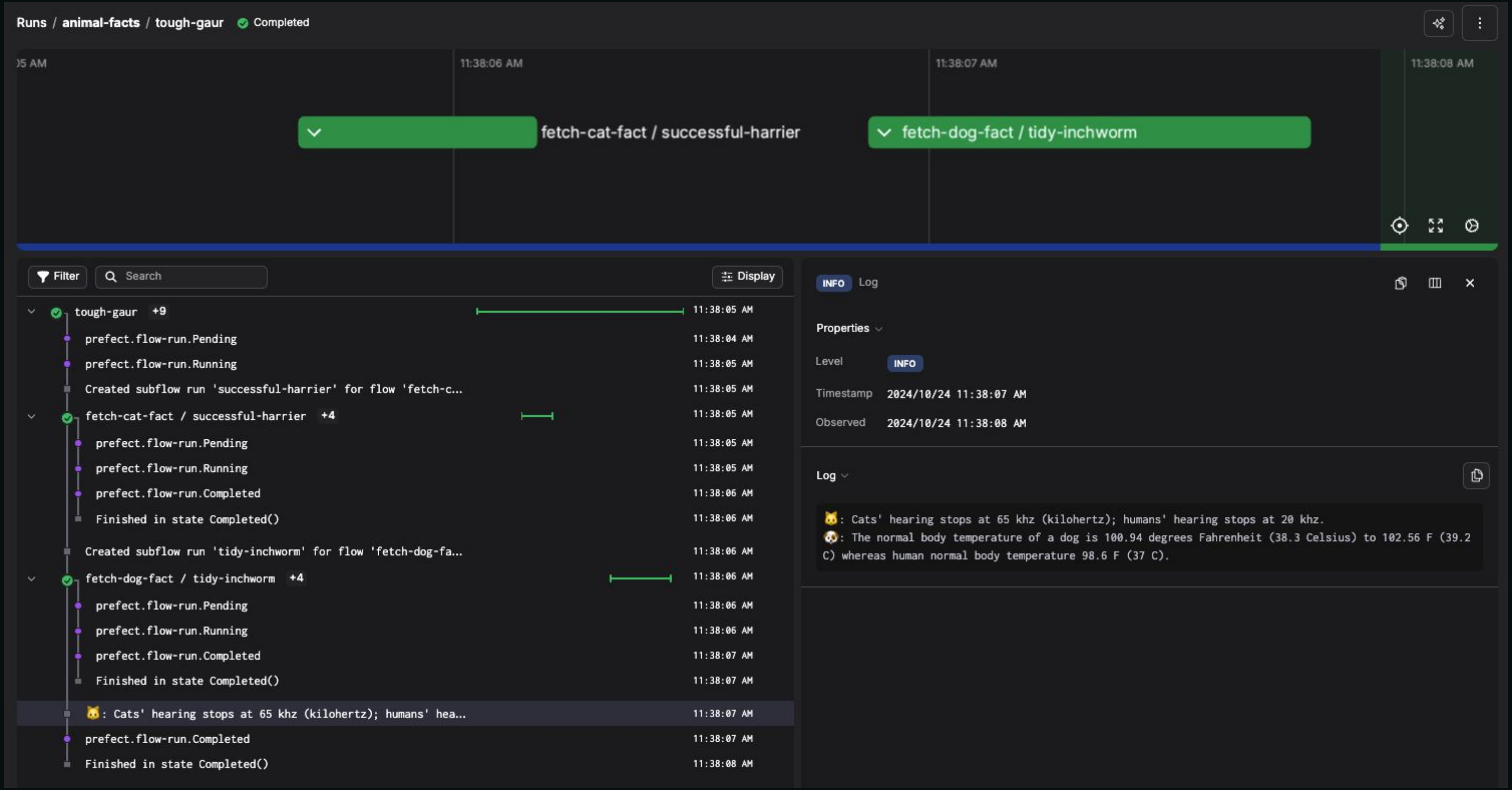


Nested flows



```python
import httpx
from prefect import flow


@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}")


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

Timeline view



Runs / animal-facts / tough-gaur Completed
5 AM
11:38:06 AM
11:38:07 AM
11:38:08 AM
fetch-cat-fact / successful-harrier
fetch-dog-fact / tidy-inchworm
Filter
Search
Display
INFO Log
Properties

```
tough-gaur +9                                                       11:38:05 AM
prefect.flow-run.Pending                                            11:38:04 AM
prefect.flow-run.Running                                            11:38:05 AM
Created subflow run 'successful-harrier' for flow 'fetch-c...       11:38:05 AM
fetch-cat-fact / successful-harrier +4                              11:38:05 AM
  prefect.flow-run.Pending                                          11:38:05 AM
  prefect.flow-run.Running                                          11:38:05 AM
  prefect.flow-run.Completed                                        11:38:06 AM
  Finished in state Completed()                                     11:38:06 AM
Created subflow run 'tidy-inchworm' for flow 'fetch-dog-fa...       11:38:06 AM
fetch-dog-fact / tidy-inchworm +4                                   11:38:06 AM
  prefect.flow-run.Pending                                          11:38:06 AM
  prefect.flow-run.Running                                          11:38:06 AM
  prefect.flow-run.Completed                                        11:38:07 AM
  Finished in state Completed()                                     11:38:07 AM
🐱: Cats' hearing stops at 65 khz (kilohertz); humans' hea...        11:38:07 AM
prefect.flow-run.Completed                                          11:38:07 AM
Finished in state Completed()                                       11:38:08 AM
```

Level INFO
Timestamp 2024/10/24 11:38:07 AM
Observed 2024/10/24 11:38:08 AM

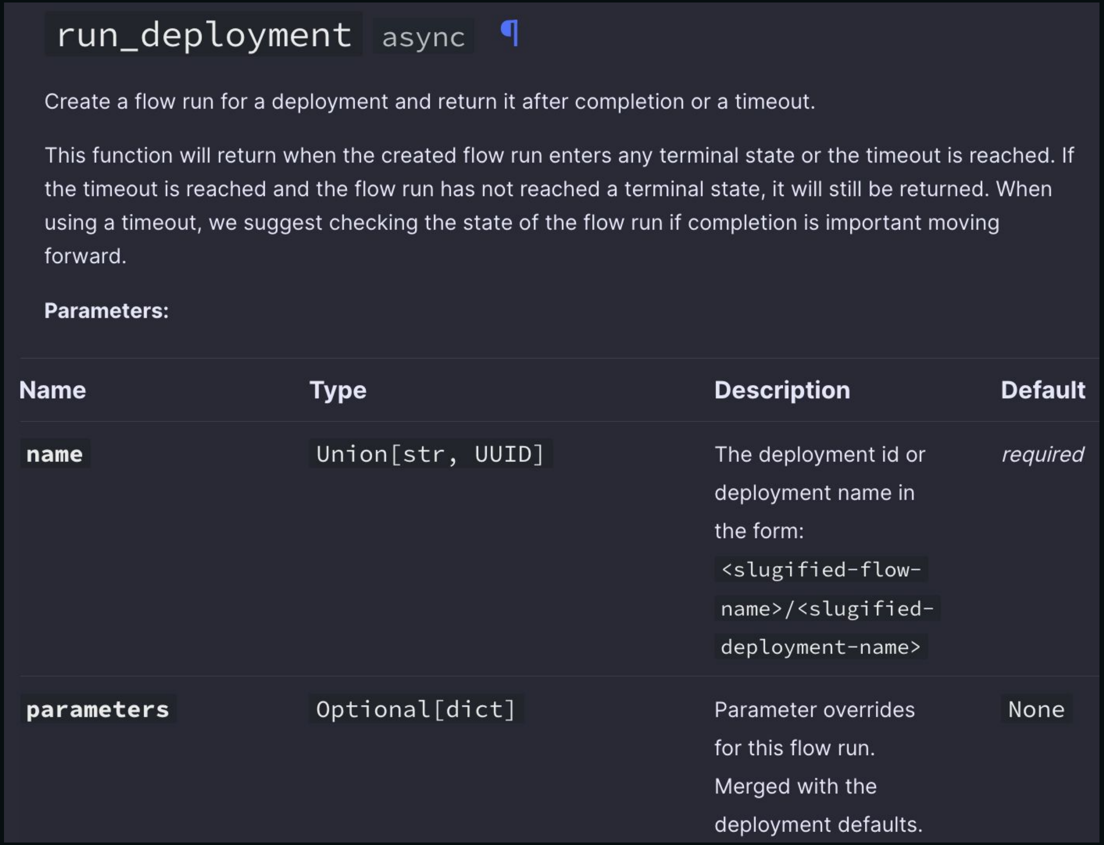
Log

```
🐱: Cats' hearing stops at 65 khz (kilohertz); humans' hearing stops at 20 khz.
🐶: The normal body temperature of a dog is 100.94 degrees Fahrenheit (38.3 Celsius) to 102.56 F (39.2
C) whereas human normal body temperature 98.6 F (37 C).
```

run_deployment

to create a flow of deployments

run_deployment



`run_deployment` `async` ¶

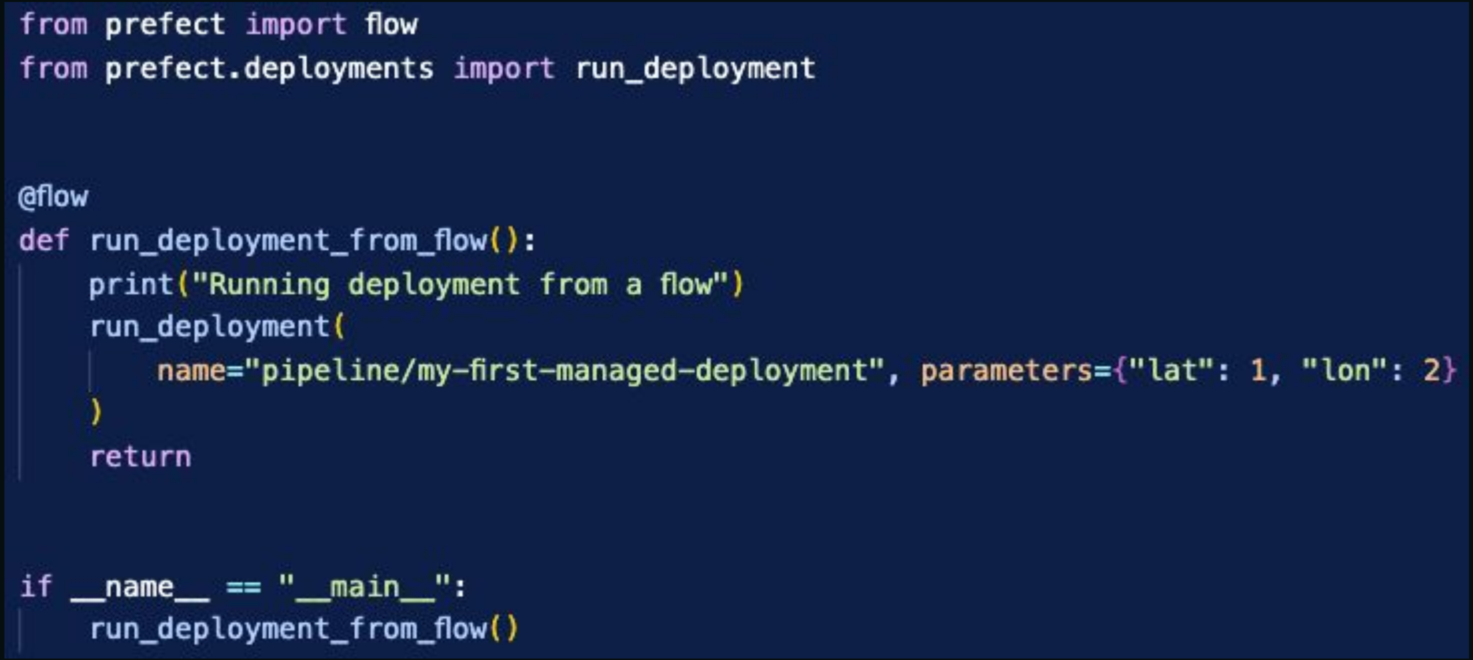
Create a flow run for a deployment and return it after completion or a timeout.

This function will return when the created flow run enters any terminal state or the timeout is reached. If the timeout is reached and the flow run has not reached a terminal state, it will still be returned. When using a timeout, we suggest checking the state of the flow run if completion is important moving forward.

Parameters:

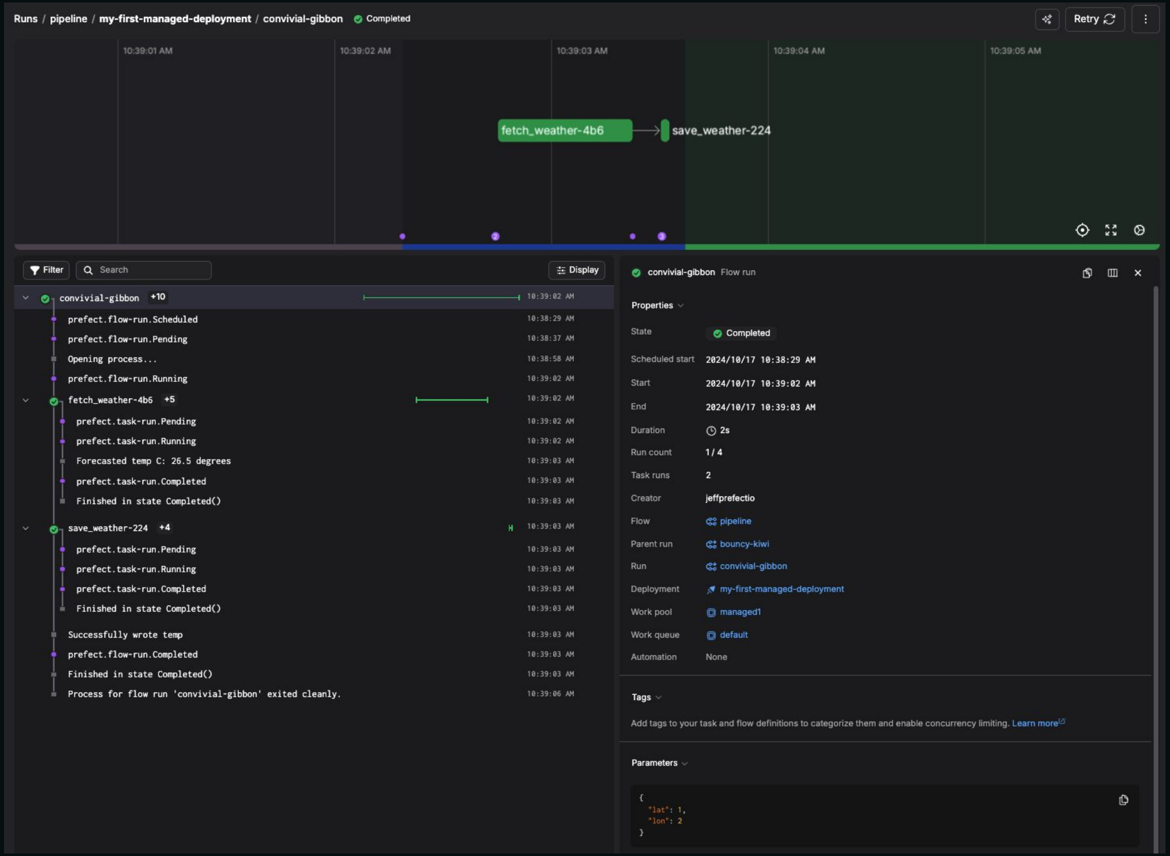
Name	Type	Description	Default
`name`	`Union[str, UUID]`	The deployment id or deployment name in the form: `<slugified-flow-name>/<slugified-deployment-name>`	*required*
`parameters`	`Optional[dict]`	Parameter overrides for this flow run. Merged with the deployment defaults.	`None`

run_deployment



```python
from prefect import flow
from prefect.deployments import run_deployment


@flow
def run_deployment_from_flow():
    print("Running deployment from a flow")
    run_deployment(
        name="pipeline/my-first-managed-deployment", parameters={"lat": 1, "lon": 2}
    )
    return


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

run_deployment



Runs / pipeline / **my-first-managed-deployment** / convivial-gibbon ✓ Completed

Retry

```
convivial-gibbon  +10                                              10:39:02 AM
  prefect.flow-run.Scheduled                                       10:38:29 AM
  prefect.flow-run.Pending                                         10:38:37 AM
  Opening process...                                               10:38:58 AM
  prefect.flow-run.Running                                         10:39:02 AM
  fetch_weather-4b6  +5                                            10:39:02 AM
    prefect.task-run.Pending                                       10:39:02 AM
    prefect.task-run.Running                                       10:39:02 AM
    Forecasted temp C: 26.5 degrees                                10:39:03 AM
    prefect.task-run.Completed                                     10:39:03 AM
    Finished in state Completed()                                  10:39:03 AM
  save_weather-224  +4                                             10:39:03 AM
    prefect.task-run.Pending                                       10:39:03 AM
    prefect.task-run.Running                                       10:39:03 AM
    prefect.task-run.Completed                                     10:39:03 AM
    Finished in state Completed()                                  10:39:03 AM
  Successfully wrote temp                                          10:39:03 AM
  prefect.flow-run.Completed                                       10:39:03 AM
  Finished in state Completed()                                    10:39:03 AM
  Process for flow run 'convivial-gibbon' exited cleanly.          10:39:06 AM
```

convivial-gibbon Flow run

Properties

State	✓ Completed
Scheduled start	2024/10/17 10:38:29 AM
Start	2024/10/17 10:39:02 AM
End	2024/10/17 10:39:03 AM
Duration	2s
Run count	1 / 4
Task runs	2
Creator	jeffprefectio
Flow	pipeline
Parent run	bouncy-kiwi
Run	convivial-gibbon
Deployment	my-first-managed-deployment
Work pool	managed1
Work queue	default
Automation	None

Tags

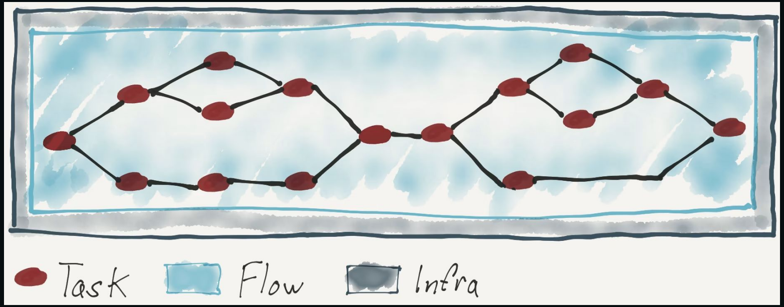
Add tags to your task and flow definitions to categorize them and enable concurrency limiting. Learn more

Parameters

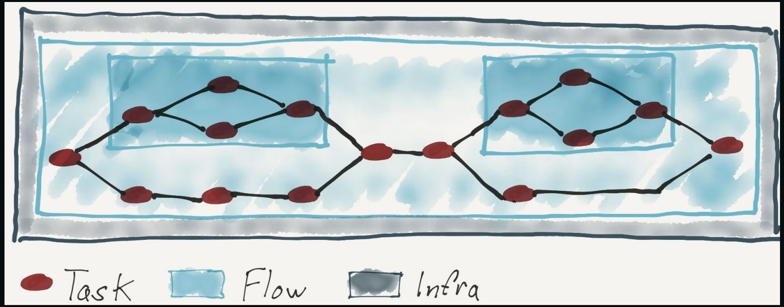
```
{
  "lat": 1,
  "lon": 2
}
```

Event-triggered workflows

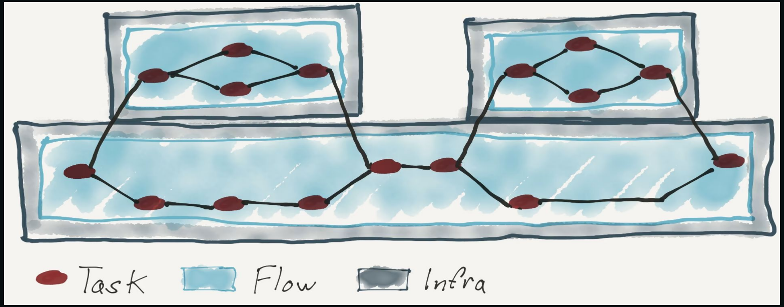
Workflow patterns - Event-triggered



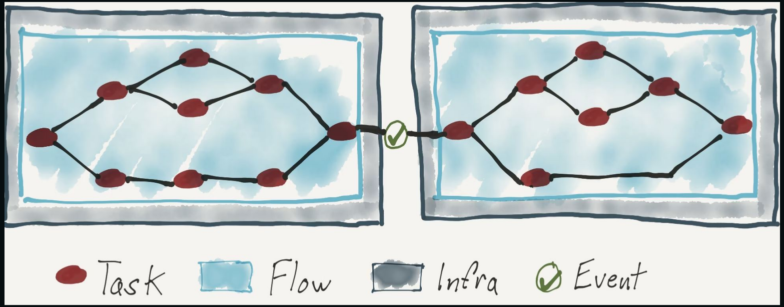
Flow of tasks



Flow of nested flows



Flow of deployments



Event-triggered flow

Event-driven flow runs with deployment triggers

Events (refresh)

Lightweight JSON bits

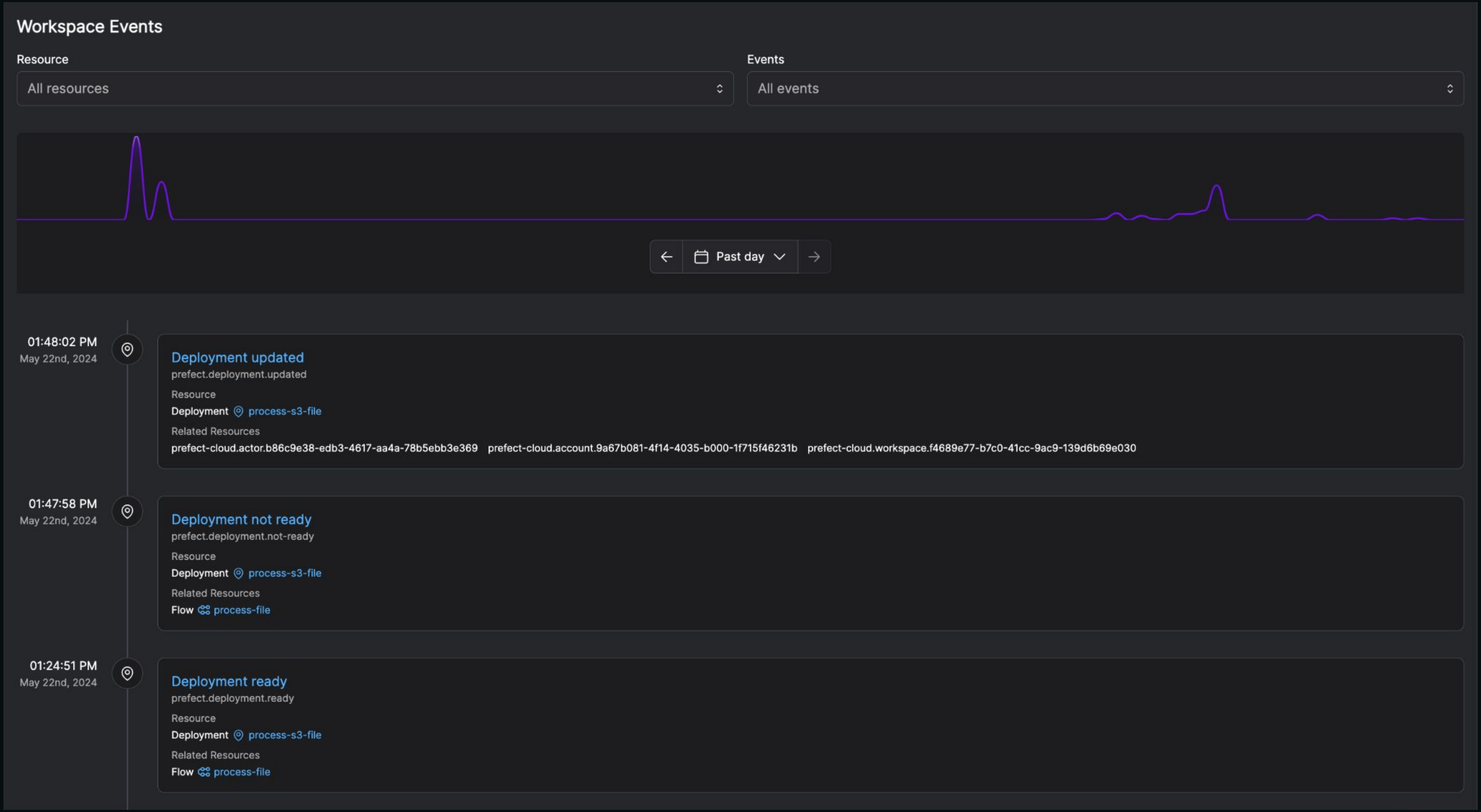
Describe what happened, who did it, etc.

Internal (Prefect-created events), examples:

- Work pool ready
- Flow run failed

External examples

- S3 object created
- Github PR opened



Workspace Events

Resource: All resources

Events: All events

01:48:02 PM
May 22nd, 2024

Deployment updated
prefect.deployment.updated
Resource
Deployment process-s3-file
Related Resources
prefect-cloud.actor.b86c9e38-edb3-4617-aa4a-78b5ebb3e369 prefect-cloud.account.9a67b081-4f14-4035-b000-1f715f46231b prefect-cloud.workspace.f4689e77-b7c0-41cc-9ac9-139d6b69e030

01:47:58 PM
May 22nd, 2024

Deployment not ready
prefect.deployment.not-ready
Resource
Deployment process-s3-file
Related Resources
Flow process-file

01:24:51 PM
May 22nd, 2024

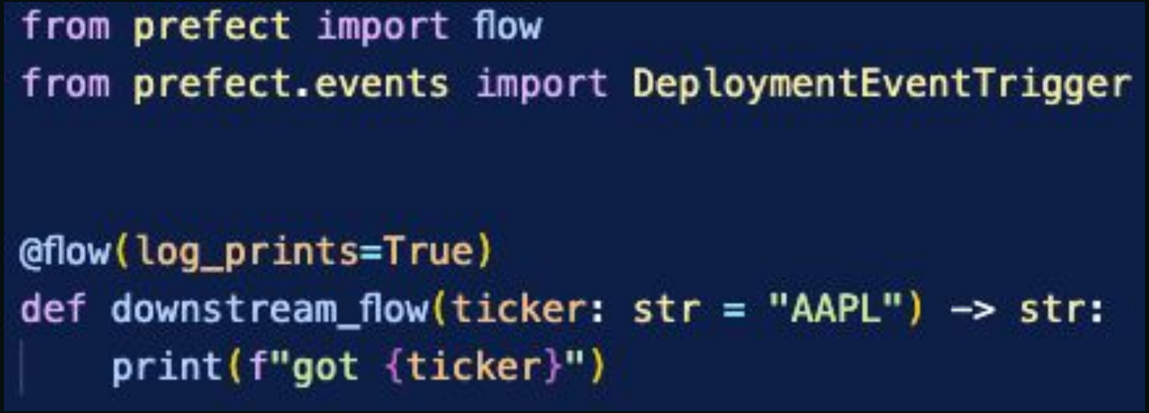
Deployment ready
prefect.deployment.ready
Resource
Deployment process-s3-file
Related Resources
Flow process-file

Deployment triggers

Allow deployments to run in response to the presence (or absence) of events

- Specify trigger condition in a *DeploymentEventTrigger* object and pass to *.deploy()*
- Linked automation created when deployment created

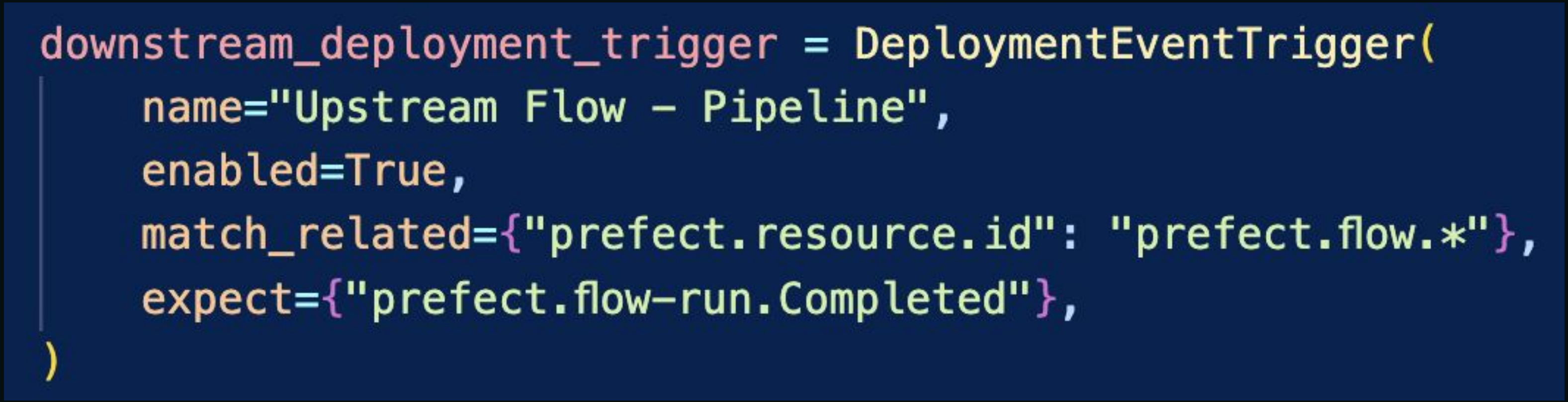
Deployment triggers - the flow to be triggered



```python
from prefect import flow
from prefect.events import DeploymentEventTrigger


@flow(log_prints=True)
def downstream_flow(ticker: str = "AAPL") -> str:
    print(f"got {ticker}")
```

Deployment triggers - the trigger

Create a *DeploymentEventTrigger* object



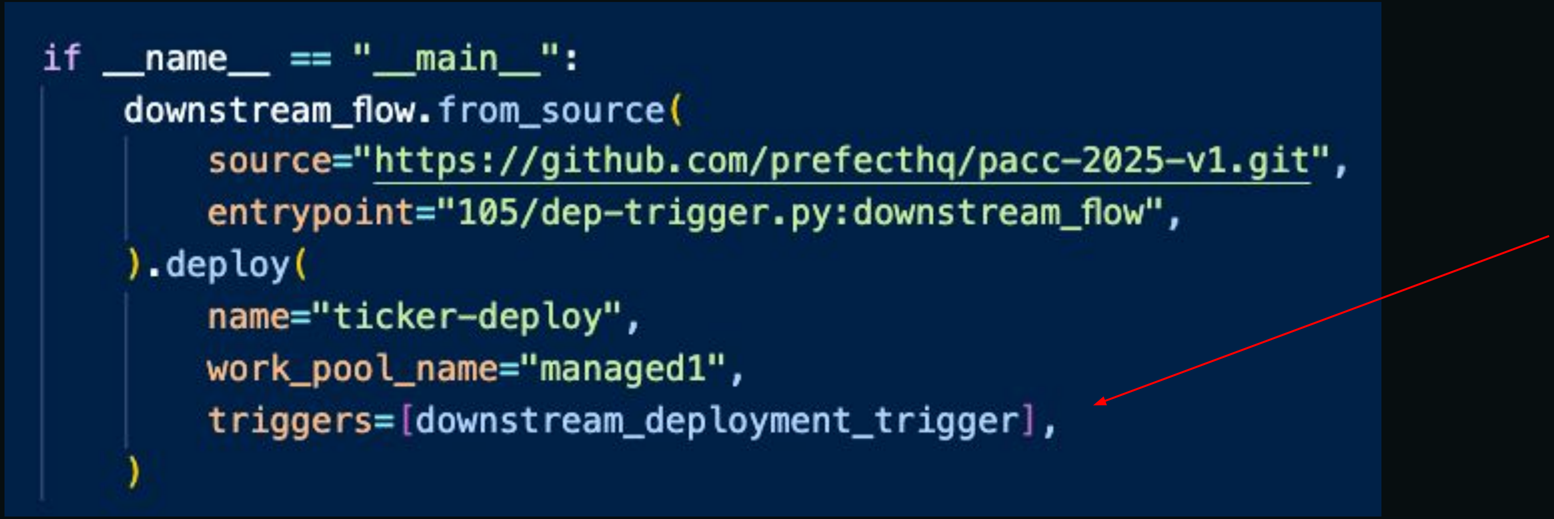
```python
downstream_deployment_trigger = DeploymentEventTrigger(
    name="Upstream Flow - Pipeline",
    enabled=True,
    match_related={"prefect.resource.id": "prefect.flow.*"},
    expect={"prefect.flow-run.Completed"},
)
```

See the event specification docs:

docs.prefect.io/latest/automate/events/events

Deployment triggers - create

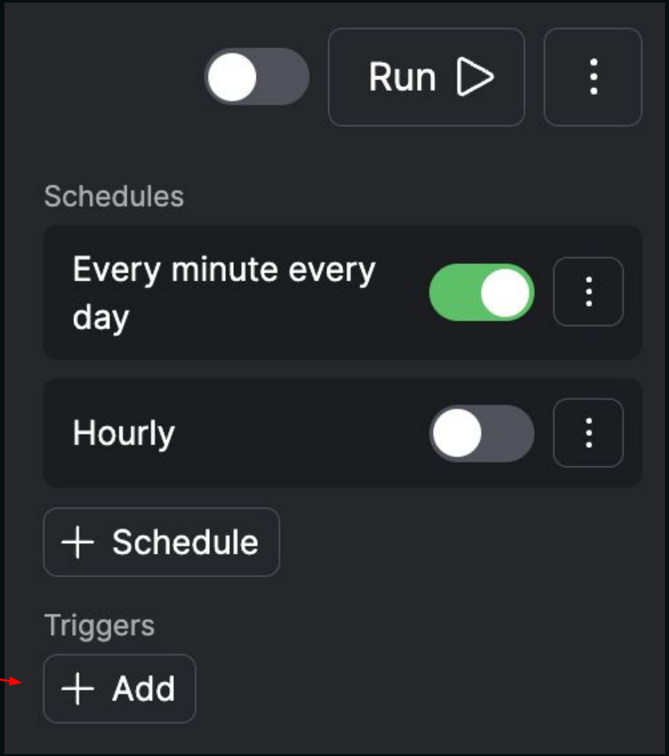
Pass the trigger object to *.deploy* and run the script



```
if __name__ == "__main__":
    downstream_flow.from_source(
        source="https://github.com/prefecthq/pacc-2025-v1.git",
        entrypoint="105/dep-trigger.py:downstream_flow",
    ).deploy(
        name="ticker-deploy",
        work_pool_name="managed1",
        triggers=[downstream_deployment_trigger],
    )
```

Another way to begin automation creation in the UI

- Start from a deployment page
- Click the **+ Add** button under **Triggers**
- Pre-populates the automation action with the deployment run

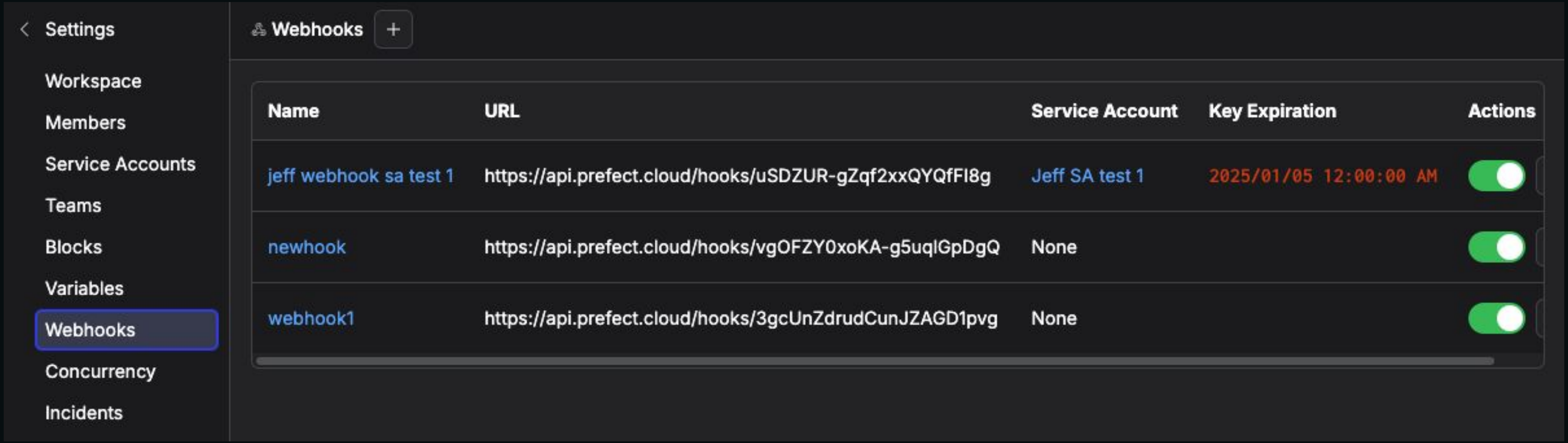


Event webhooks 🕸️ 🪝

Event webhooks

- Exposes URL endpoint
- Interface for integrating external apps
- When webhook URL pinged, creates Prefect event
  - Can be used as automation trigger
- Great when **not** in Python land

Event webhooks



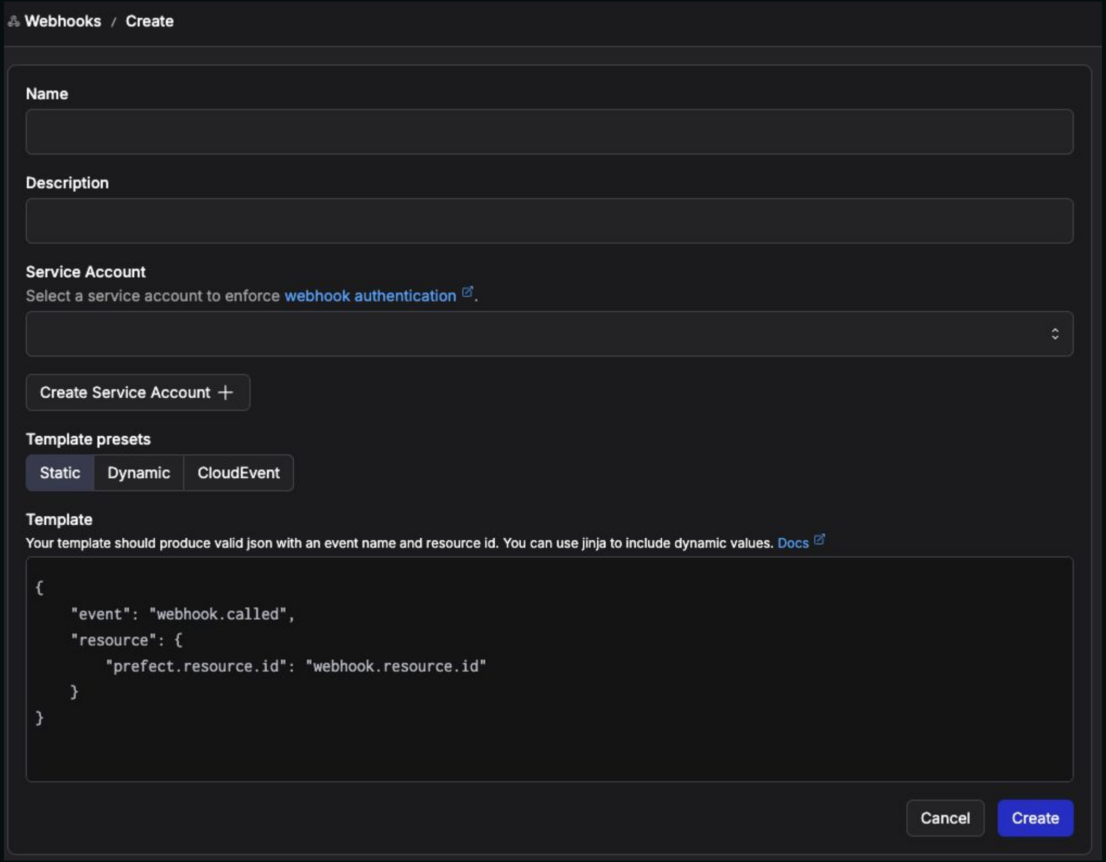
Settings

- Workspace
- Members
- Service Accounts
- Teams
- Blocks
- Variables
- Webhooks
- Concurrency
- Incidents

Webhooks +

Name	URL	Service Account	Key Expiration	Actions
jeff webhook sa test 1	https://api.prefect.cloud/hooks/uSDZUR-gZqf2xxQYQfFI8g	Jeff SA test 1	2025/01/05 12:00:00 AM	🟩
newhook	https://api.prefect.cloud/hooks/vgOFZY0xoKA-g5uqlGpDgQ	None		🟩
webhook1	https://api.prefect.cloud/hooks/3gcUnZdrudCunJZAGD1pvg	None		🟩

Event webhooks


Webhooks / Create

Name

Description

Service Account
Select a service account to enforce webhook authentication.

Template presets

Template
Your template should produce valid json with an event name and resource id. You can use jinja to include dynamic values. Docs

```
{
    "event": "webhook.called",
    "resource": {
        "prefect.resource.id": "webhook.resource.id"
    }
}
```

Event webhooks

- Use Jinja2 for dynamic templating
- Template must be valid JSON
- Create from UI or CLI
- Prefect Cloud Pro tier has authentication option through Service Accounts

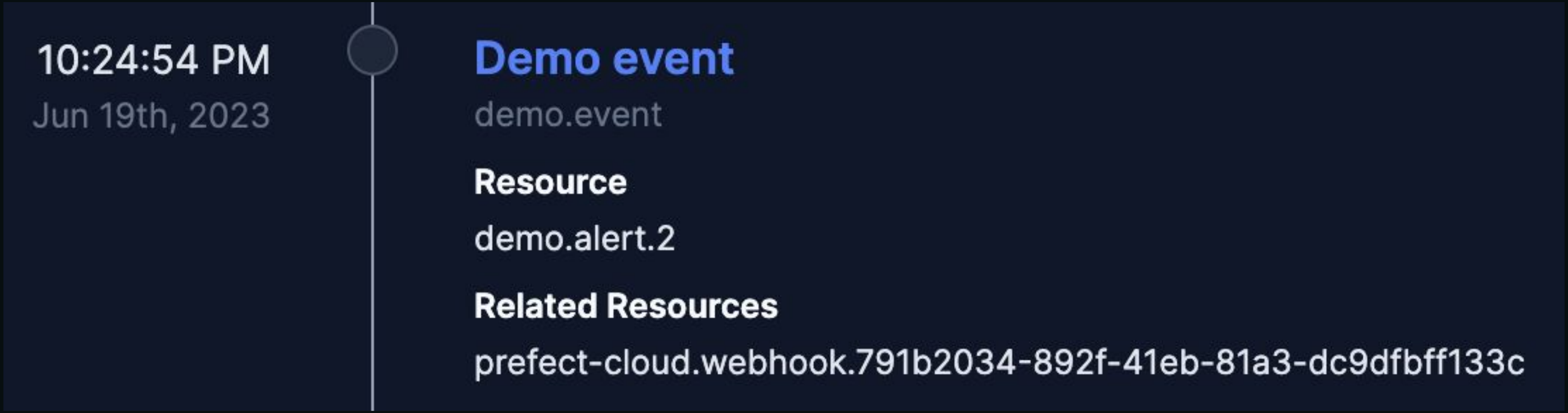
Event webhooks

Hit the endpoint provided by Prefect:

curl https://api.prefect.cloud/hooks/your_slug_here

Event webhooks

See new event on **Event Feed** tab in the UI



10:24:54 PM
Jun 19th, 2023

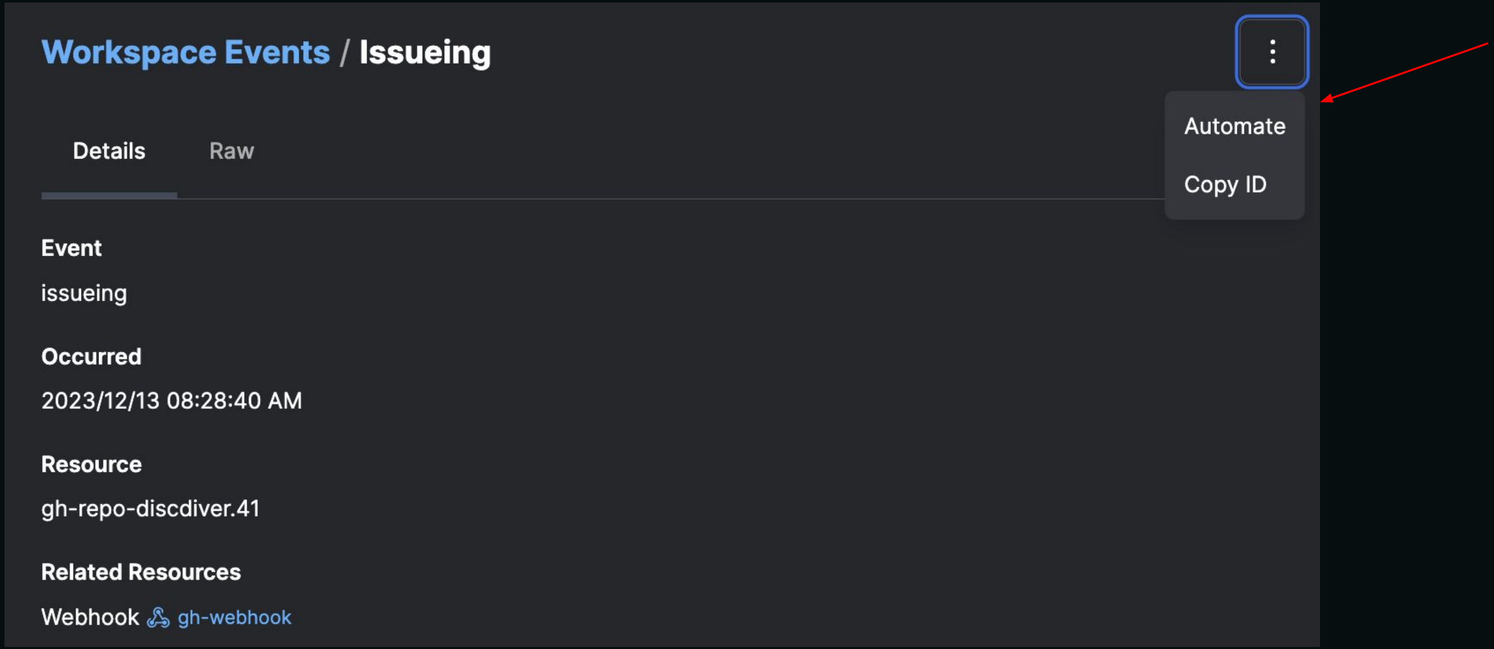
Demo event
demo.event

Resource
demo.alert.2

Related Resources
prefect-cloud.webhook.791b2034-892f-41eb-81a3-dc9dfbff133c

Event webhooks

⚡ Use this event as a custom trigger in an automation!



Workspace Events / Issueing

Event
issueing

Occurred
2023/12/13 08:28:40 AM

Resource
gh-repo-discdiver.41

Related Resources
Webhook gh-webhook

Event webhooks - example

When an object lands in an S3 bucket, can use EventBridge or Lambda to hit a Prefect webhook you've created.

Example:

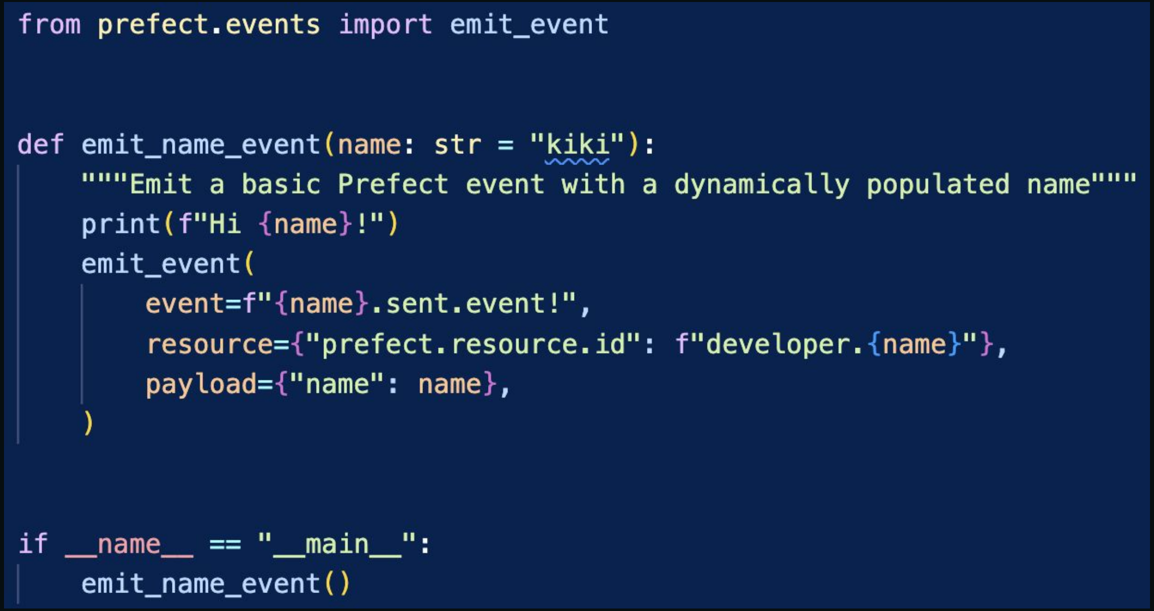
github.com/PrefectHQ/prefect-demos/blob/main/flows/aws/datalake/README.md

Custom events defined in Python 🐍

Create custom event to be emitted when code runs

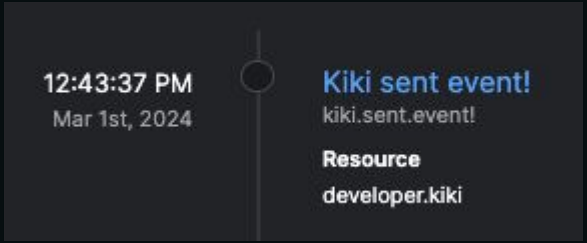
emit_event requires two args:

event and *resource={":prefect.resource.id: val"}*

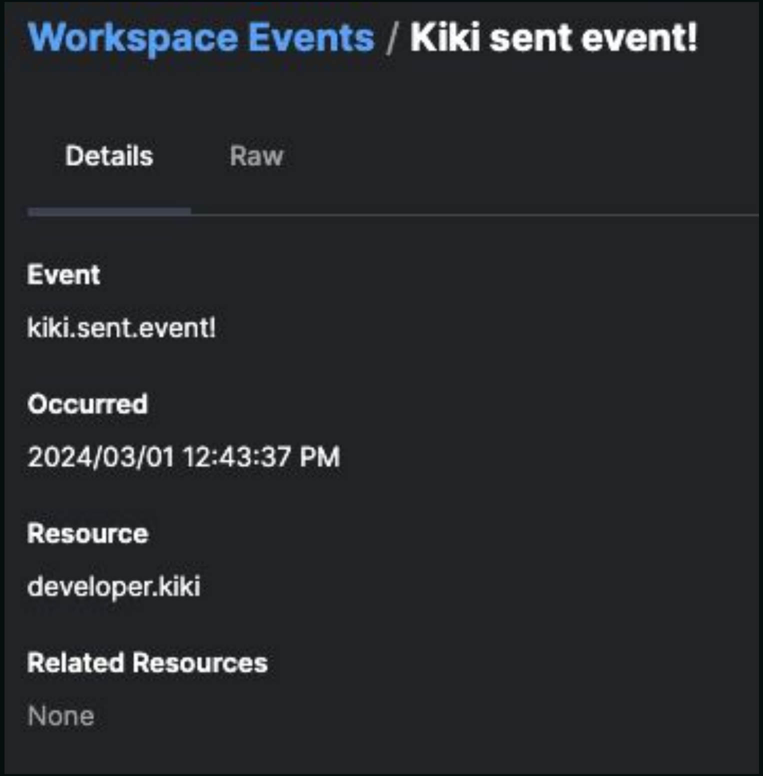


```python
from prefect.events import emit_event


def emit_name_event(name: str = "kiki"):
    """Emit a basic Prefect event with a dynamically populated name"""
    print(f"Hi {name}!")
    emit_event(
        event=f"{name}.sent.event!",
        resource={"prefect.resource.id": f"developer.{name}"},
        payload={"name": name},
    )


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

Run code and head to the **Events** page


12:43:37 PM
Mar 1st, 2024
Kiki sent event!
kiki.sent.event!
Resource
developer.kiki

Click link to see event page


Workspace Events / Kiki sent event!

Details Raw

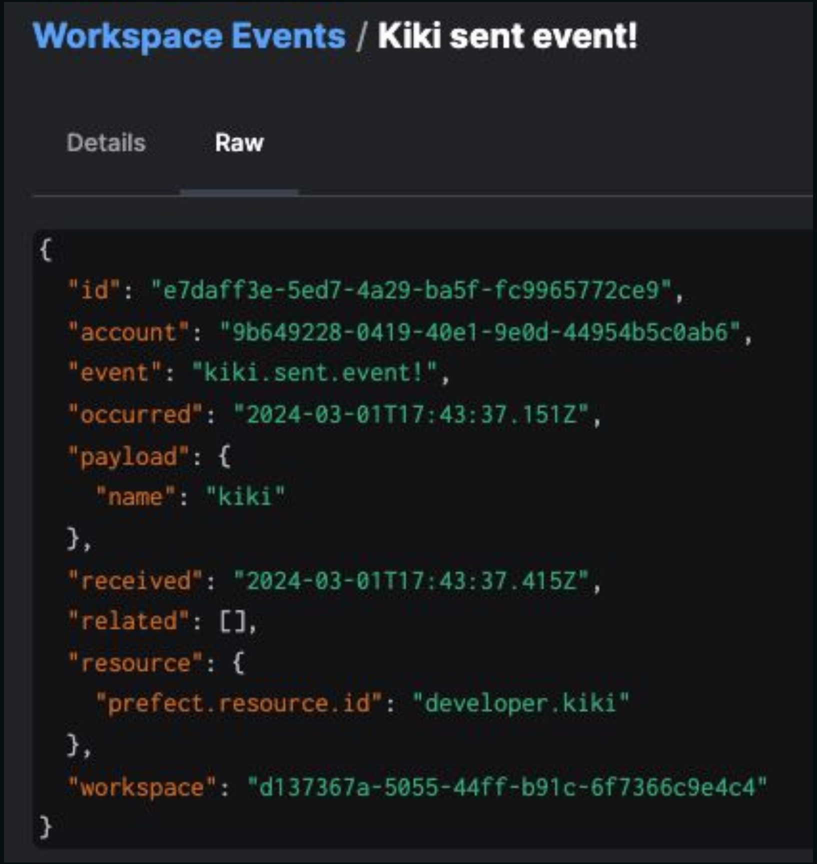
Event
kiki.sent.event!

Occurred
2024/03/01 12:43:37 PM

Resource
developer.kiki

Related Resources
None

See event details on the **Raw** tab



```
{
  "id": "e7daff3e-5ed7-4a29-ba5f-fc9965772ce9",
  "account": "9b649228-0419-40e1-9e0d-44954b5c0ab6",
  "event": "kiki.sent.event!",
  "occurred": "2024-03-01T17:43:37.151Z",
  "payload": {
    "name": "kiki"
  },
  "received": "2024-03-01T17:43:37.415Z",
  "related": [],
  "resource": {
    "prefect.resource.id": "developer.kiki"
  },
  "workspace": "d137367a-5055-44ff-b91c-6f7366c9e4c4"
}
```

Data from event can be used in an automation action

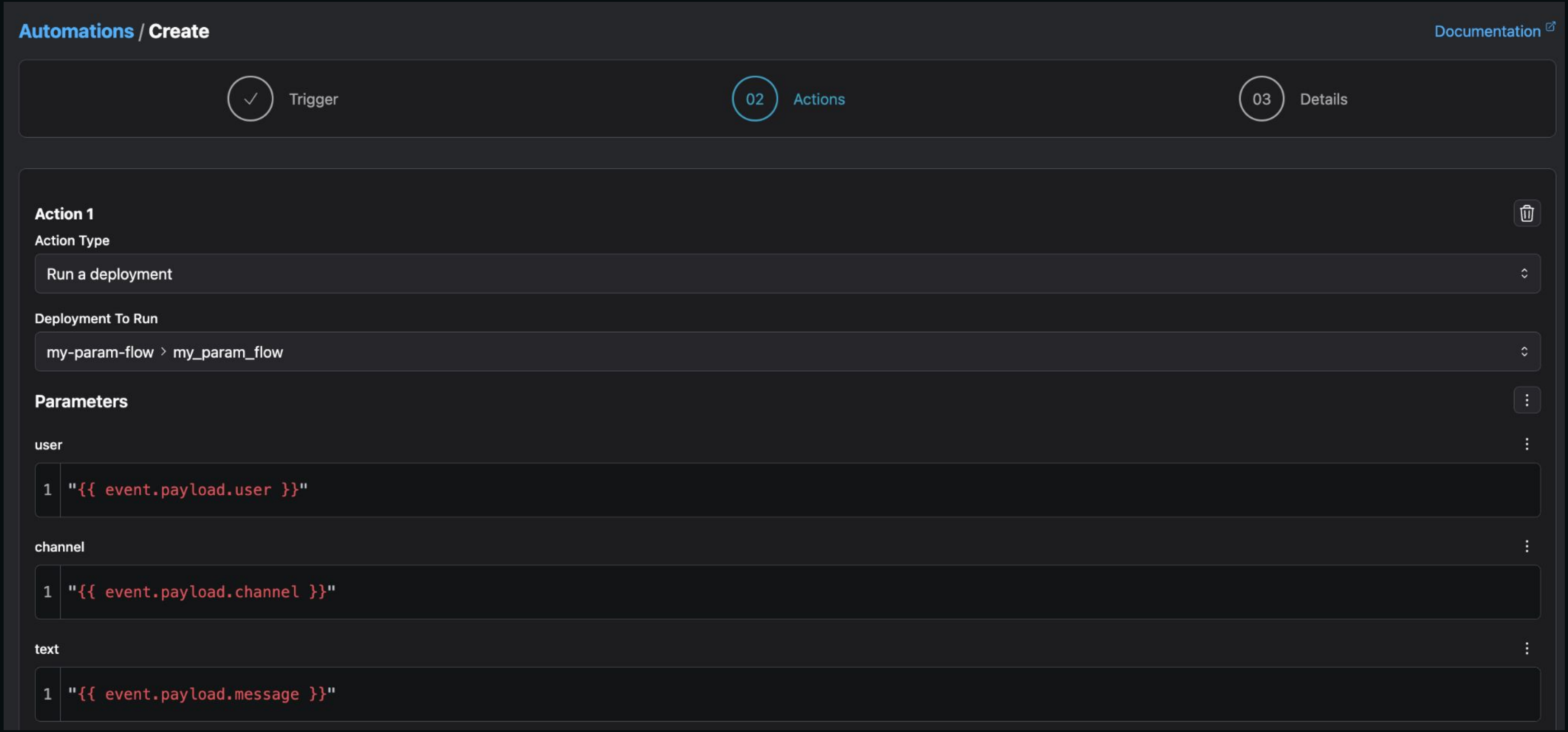
For example: Populate a flow param via a *Run Deployment* action

Use *emit_event*'s *payload* parameter

Example: custom event with detailed payload

```
from prefect.events import emit_event

emit_event(
    event=f"bot.{bot.name.lower()}.responded",
    resource={"prefect.resource.id": f"bot.{bot.name.lower()}"},
    payload={
        "user": event.user,
        "channel": event.channel,
        "thread_ts": thread,
        "text": text,
        "response": response.content,
        "prompt_tokens": prompt_tokens,
        "response_tokens": response_tokens,
        "total_tokens": prompt_tokens + response_tokens,
    },
)
```

Use payload data in event-driven flow runs



Automations / Create

Documentation

✓ Trigger

02 Actions

03 Details

Action 1

Action Type

Run a deployment

Deployment To Run

my-param-flow > my_param_flow

Parameters

user

```
"{{ event.payload.user }}"
```

channel

```
"{{ event.payload.channel }}"
```

text

```
"{{ event.payload.message }}"
```

Advice

- Use a flow with tasks for data engineering use cases unless you need another solution
- Use *run_deployment* if need to run a flow on different infrastructure
- Use event-driven workflows if want to run in response to an event

Tasks ++ 💪

Tasks on their own

- Lightweight
- Celery replacement
- Can run in background
- Can be nested and run outside of flows
- Can be parallelized
- Run client side; info sent to the API in batches
- Fast, but less real-time info available in UI

Note: you need a flow to create a deployment

docs.prefect.io/latest/develop/deferred-tasks

105 Recap

You’ve seen how to use several workflow patterns with

- Nested flows
- *run_deployment*
- Event-based automations
    - Deployment event triggers defined in Python
    - Webhooks
    - Custom events defined in Python

- Tasks alone

105 Lab

- Create a deployment with nested flows and run it.
- Create a flow that uses *run_deployment* to run another deployment.
- Stretch 1: Create a custom event in Python that triggers a notification action in an automation.
- Stretch 2: Create a webhook. Create an automation that pauses work when the webhook fires.