# 104 - Easily switch infrastructure and manage teams with work pool-based deployments

PREFECT

## 104 Agenda

- Create work pool-based deployments with *.deploy()*
- Run flows on Prefect's infrastructure with a Prefect Managed work pool
- Use a worker with a hybrid work pool for maximum control
  - Process
  - Docker
- Store your flow code
  - On GitHub
  - In a Docker image

## Why use a work pool-based deployment?

Infrastructure is a pain, Prefect makes it better. 🙂

- Run workflows on a variety of dynamic infrastructure
- Provide a template for teams
- Scale infrastructure to 0 (serverless)
- Prioritize work

## Create deployment with *.deploy()*

Very similar syntax to *.serve()*

Differences:

- Must specify work pool
- Must specify flow code storage source (or Docker image)
- Does **not** start a process watching for scheduled runs

## Managed work pools 🤖

## Prefect Managed work pools

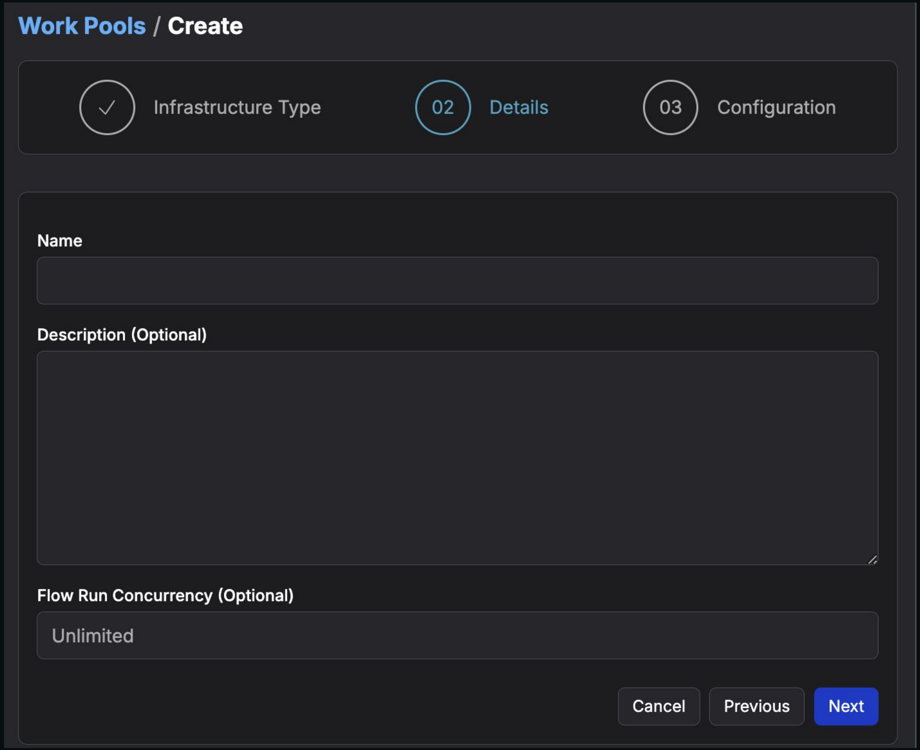
- Run workflows on Prefect’s infrastructure
- Cloud only
- Easy mode - no worker required
- Limitations
    - Compute hours
    - Concurrency
    - No custom Docker image

## First work pool-based deployment

- Create deployment with *.deploy()*
- Specify flow code stored in a GitHub repository with *.from_source()*
- Use a **Prefect Managed** work pool

# Create a Prefect Managed work pool

In the UI, **Work Pools -> + -> Prefect Managed**


Work Pools / Create

Infrastructure Type | 02 Details | 03 Configuration

Name

Description (Optional)

Flow Run Concurrency (Optional)

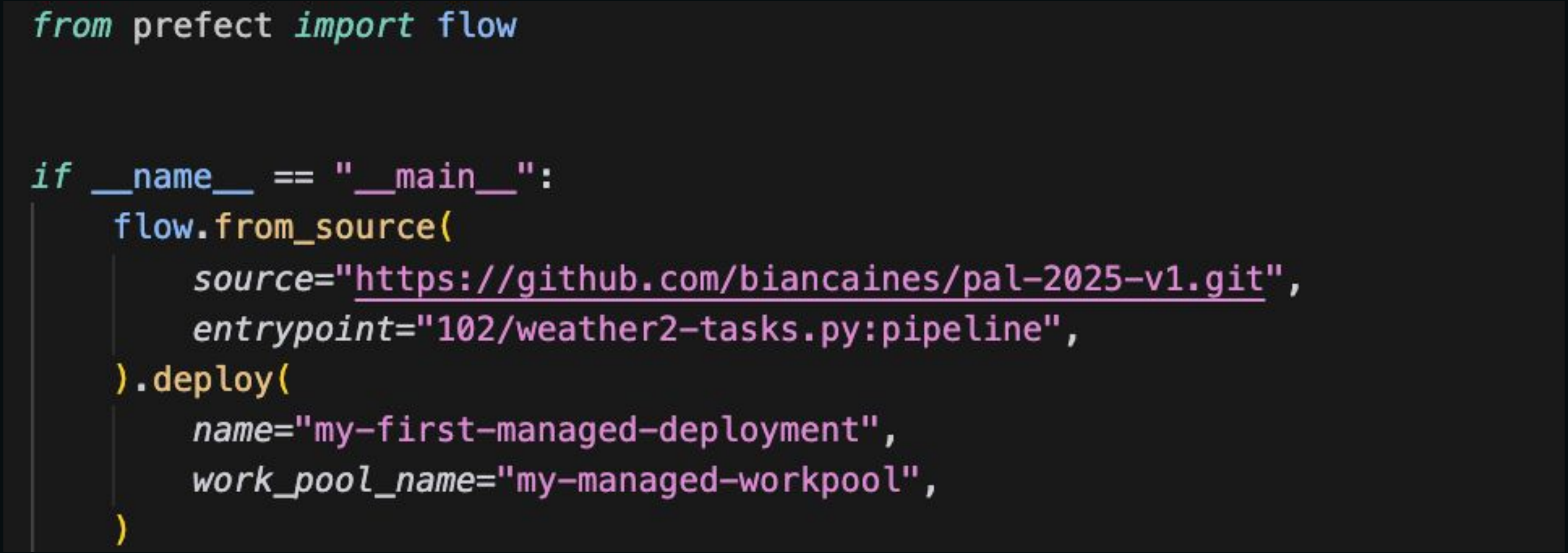
Unlimited

Cancel | Previous | Next

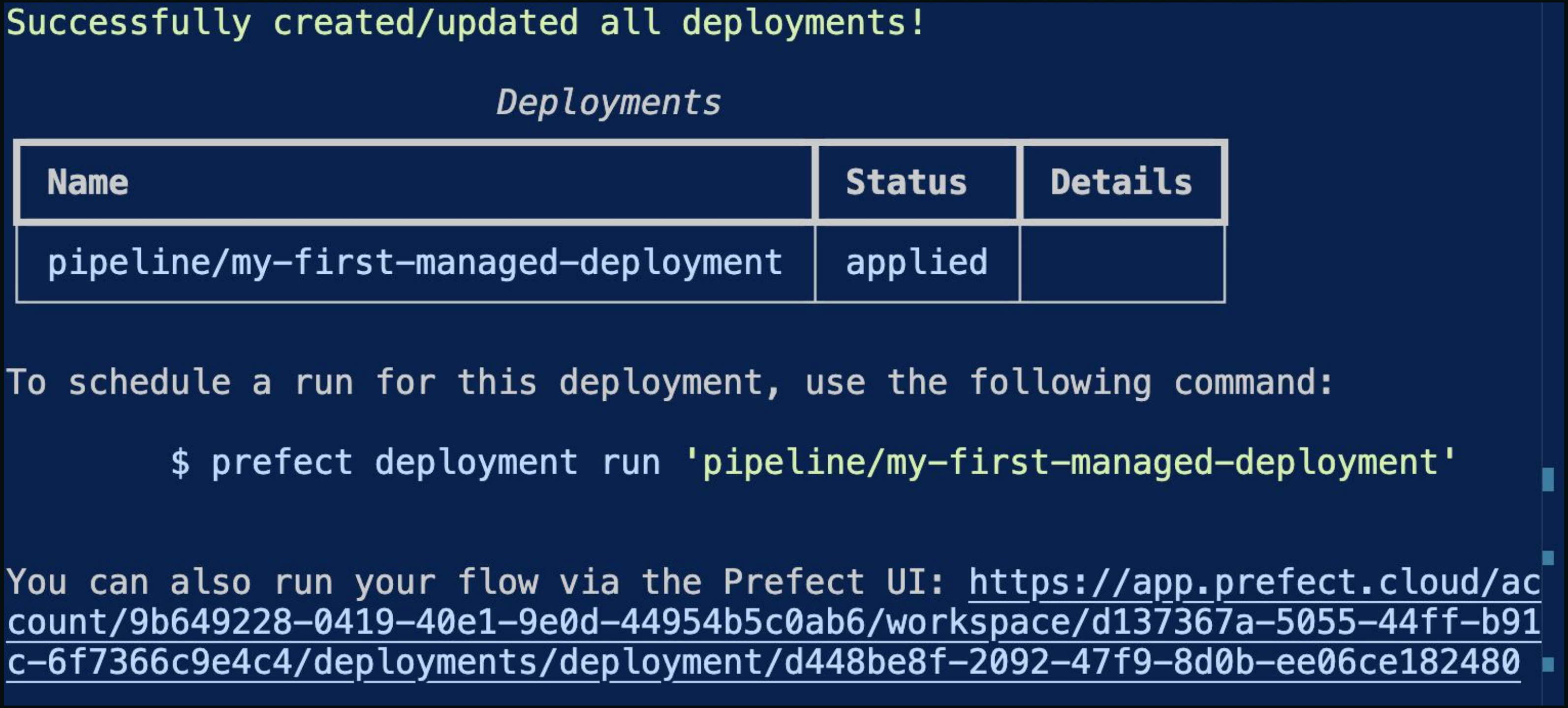
## Work pools

- Don’t modify the job template for now
- You can specify environment variables, etc.
- Work pools make it easier for data engineering platform teams to create guardrails for other teams

## Deployment with Prefect Managed work pool

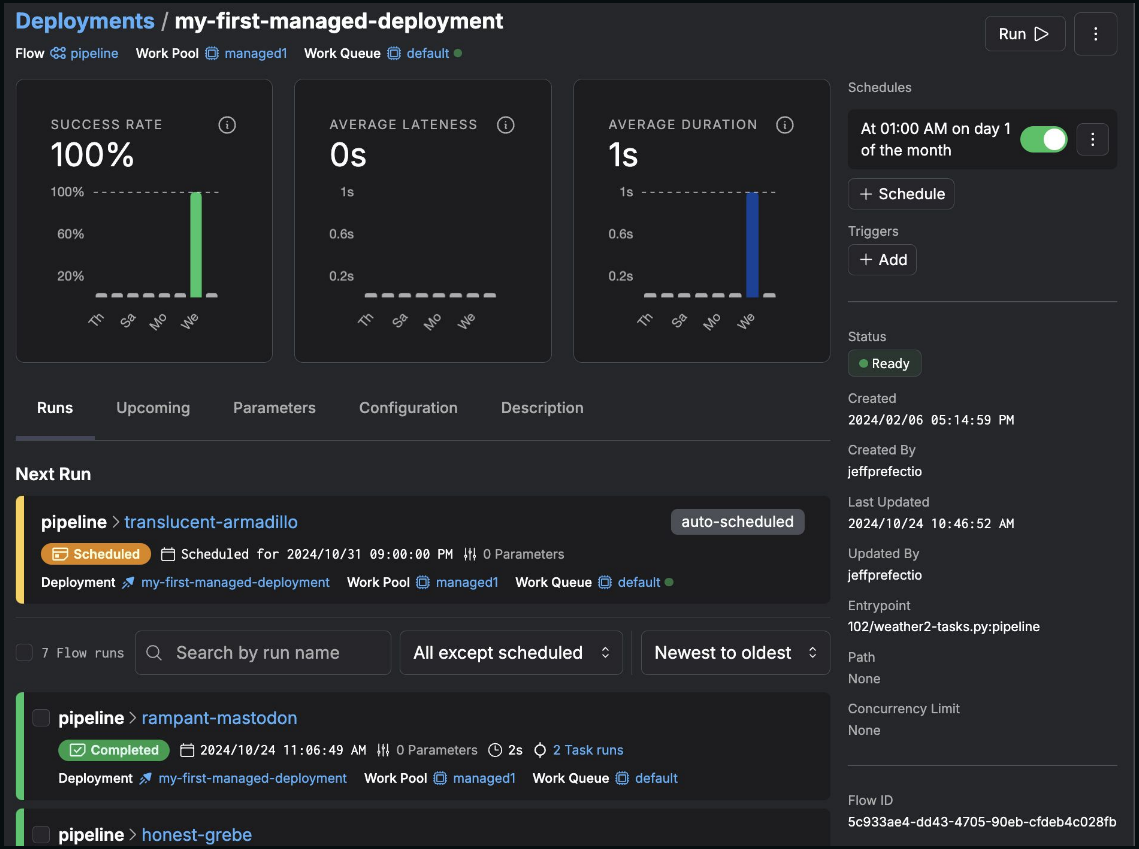

```python
from prefect import flow


if __name__ == "__main__":
    flow.from_source(
        source="https://github.com/biancaines/pal-2025-v1.git",
        entrypoint="102/weather2-tasks.py:pipeline",
    ).deploy(
        name="my-first-managed-deployment",
        work_pool_name="my-managed-workpool",
    )
```

## Run script to create the deployment


```
Successfully created/updated all deployments!

                        Deployments
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━┳━━━━━━━━━┓
┃ Name                                  ┃ Status  ┃ Details ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━╇━━━━━━━━━┩
│ pipeline/my-first-managed-deployment  │ applied │         │
└───────────────────────────────────────┴─────────┴─────────┘

To schedule a run for this deployment, use the following command:

        $ prefect deployment run 'pipeline/my-first-managed-deployment'


You can also run your flow via the Prefect UI: https://app.prefect.cloud/ac
count/9b649228-0419-40e1-9e0d-44954b5c0ab6/workspace/d137367a-5055-44ff-b91
c-6f7366c9e4c4/deployments/deployment/d448be8f-2092-47f9-8d0b-ee06ce182480
```

# See deployment details in the UI

1.



**Deployments / my-first-managed-deployment**

Flow pipeline Work Pool managed1 Work Queue default

SUCCESS RATE 100%

AVERAGE LATENESS 0s

AVERAGE DURATION 1s

Schedules

At 01:00 AM on day 1 of the month

Triggers

Status
Ready

Created
2024/02/06 05:14:59 PM

Created By
jeffprefectio

Last Updated
2024/10/24 10:46:52 AM

Updated By
jeffprefectio

Entrypoint
102/weather2-tasks.py:pipeline

Path
None

Concurrency Limit
None

Flow ID
5c933ae4-dd43-4705-90eb-cfdeb4c028fb

**Next Run**

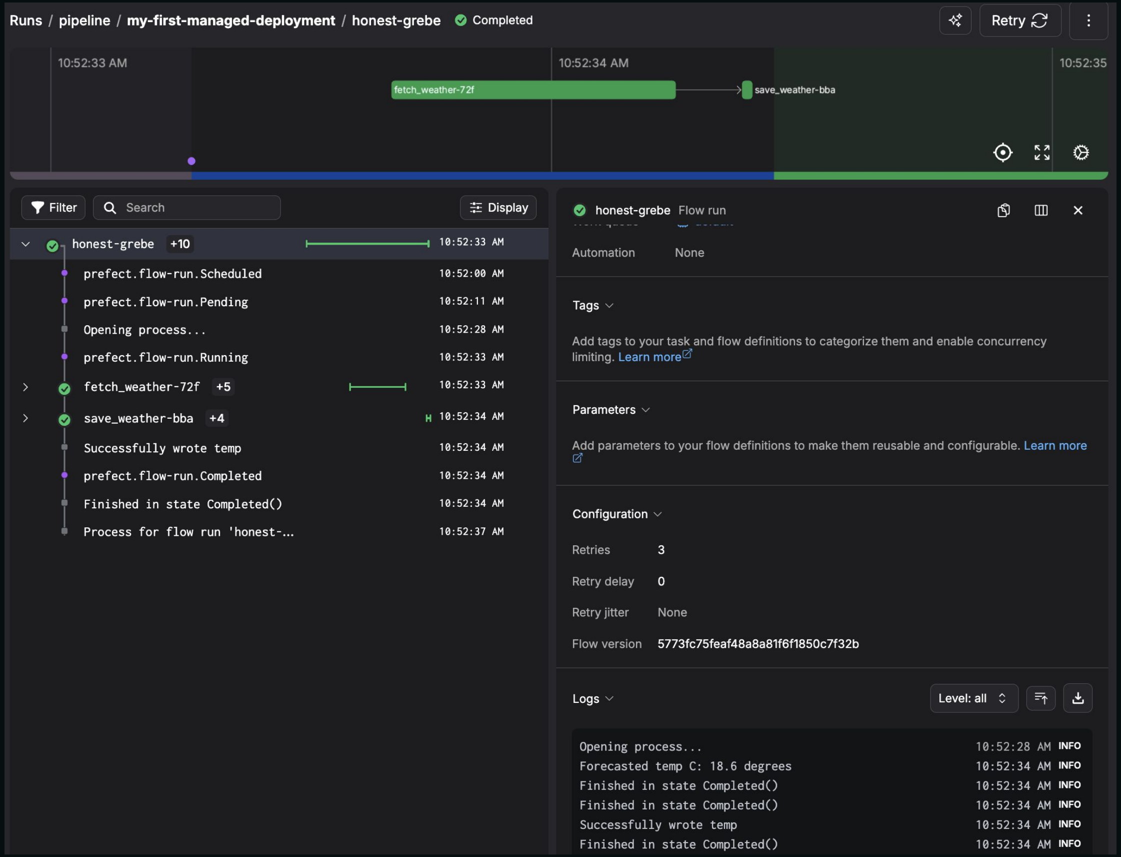
pipeline > translucent-armadillo auto-scheduled
Scheduled Scheduled for 2024/10/31 09:00:00 PM 0 Parameters
Deployment my-first-managed-deployment Work Pool managed1 Work Queue default

7 Flow runs | Search by run name | All except scheduled | Newest to oldest

pipeline > rampant-mastodon
Completed 2024/10/24 11:06:49 AM 0 Parameters 2s 2 Task runs
Deployment my-first-managed-deployment Work Pool managed1 Work Queue default

pipeline > honest-grebe

# Run the deployment



Runs / pipeline / my-first-managed-deployment / honest-grebe  Completed

```
honest-grebe +10                                   10:52:33 AM
  prefect.flow-run.Scheduled                       10:52:00 AM
  prefect.flow-run.Pending                         10:52:11 AM
  Opening process...                               10:52:28 AM
  prefect.flow-run.Running                         10:52:33 AM
  fetch_weather-72f +5                             10:52:33 AM
  save_weather-bba +4                              10:52:34 AM
  Successfully wrote temp                          10:52:34 AM
  prefect.flow-run.Completed                       10:52:34 AM
  Finished in state Completed()                    10:52:34 AM
  Process for flow run 'honest-...                 10:52:37 AM
```

honest-grebe Flow run

Automation None

Tags

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

Parameters

Add parameters to your flow definitions to make them reusable and configurable. Learn more

Configuration

| | |
|---|---|
| Retries | 3 |
| Retry delay | 0 |
| Retry jitter | None |
| Flow version | 5773fc75feaf48a8a81f6f1850c7f32b |

Logs  Level: all

```
Opening process...                         10:52:28 AM INFO
Forecasted temp C: 18.6 degrees            10:52:34 AM INFO
Finished in state Completed()              10:52:34 AM INFO
Finished in state Completed()              10:52:34 AM INFO
Successfully wrote temp                    10:52:34 AM INFO
Finished in state Completed()              10:52:34 AM INFO
```

## At runtime, Prefect:

1. Pulls your flow code from GitHub
2. Runs your code in a Docker container on our infrastructure
3. Monitors and reports on state
4. Exits container and cleans up

## Run the deployment

- Run state progression:
  *Scheduled -> Pending -> Running -> Completed*

- ⌚ Takes a moment to spin up Docker Container on our infrastructure

## Let’s break this down

## Work pools

## Work pools

- Server side
- Provide default infrastructure configuration for deployments
- **👆Deployments that use this work pool inherit these settings**

# Flow code storage

## Flow code storage options

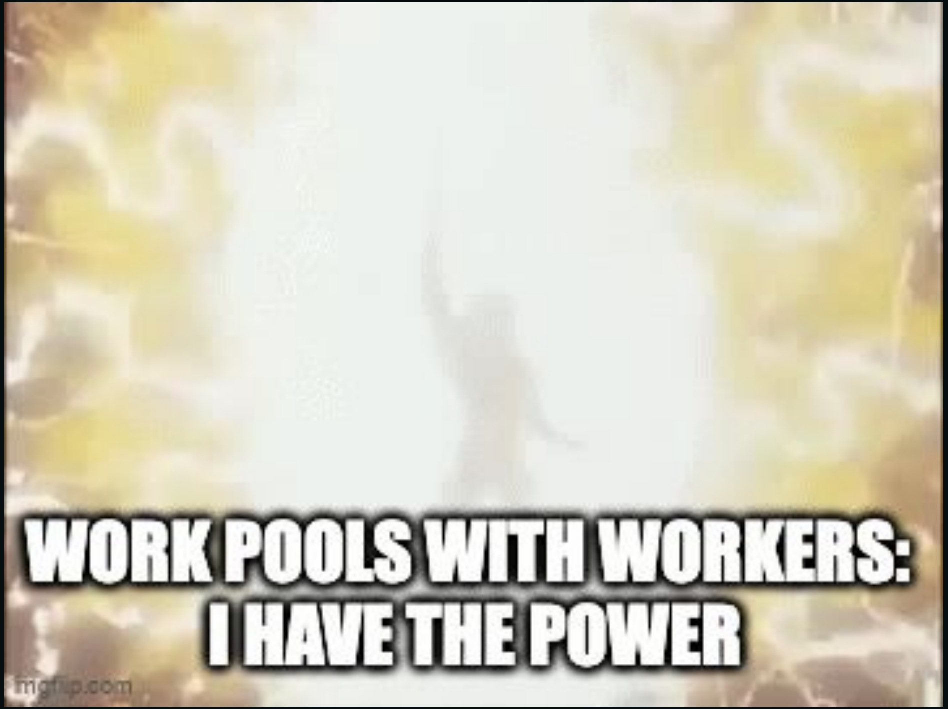
1. Local
2. Git-based remote repository (e.g. GitHub, GitLab)
3. Bake your code into a Docker image
4. Cloud provider storage

## Flow code storage

- Specified public GitHub repo with *.from_source()* class method
- Call *flow.from_source()* or *flow_name.from_source()*
- Provide repo URL and *entrypoint path:flow function name*
- 🔐 Private repos are fine, just pass credentials

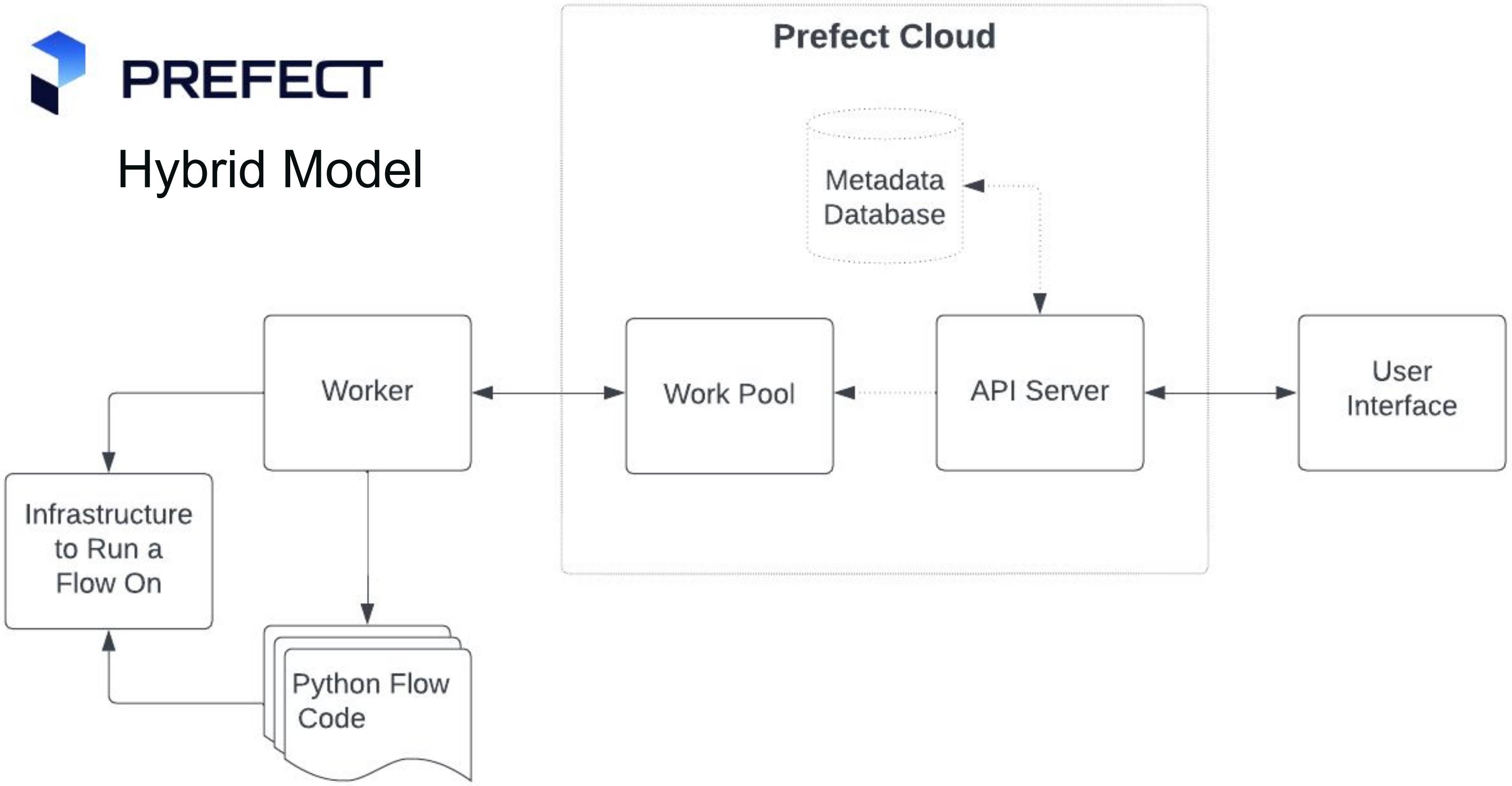
## Hybrid model - hybrid work pools with workers

## Hybrid work pools with workers



PREFECT

# Hybrid Model



## Hybrid model = separation

- Your flow code runs on your infrastructure
- Your flow code is stored on your storage (GitLab, GitHub, AWS, Docker image, etc)
- Prefect Cloud stores metadata, logs, artifacts, etc.
- Data encrypted at rest
- Prefect Technologies, Inc. is SOC2 Type II compliant

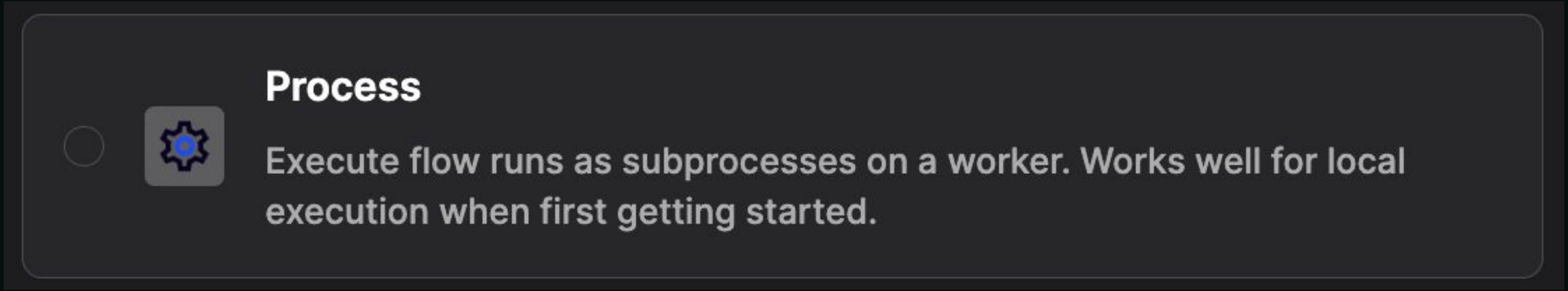
https://www.prefect.io/security

## Example: Process work pool & worker

## First hybrid work pool-based deployment

- Create with *.deploy()*
- Specify flow code stored in a GitHub repository with *.from_source()*
- Use a **Process** work pool
- Start a worker to pick up scheduled flow runs ⭐

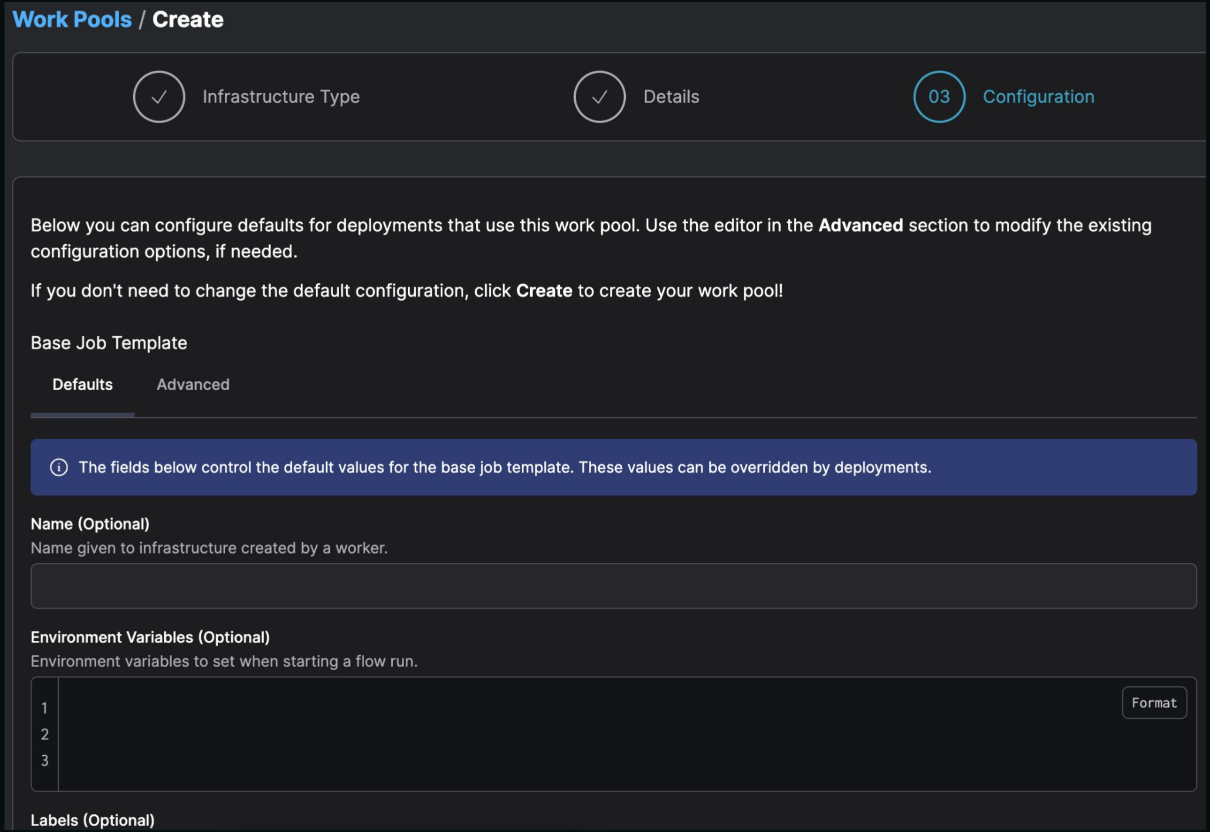
## Create a Process work pool



**Process**

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

# Create a Process work pool



Work Pools / Create

Infrastructure Type — Details — 03 Configuration

Below you can configure defaults for deployments that use this work pool. Use the editor in the **Advanced** section to modify the existing configuration options, if needed.

If you don't need to change the default configuration, click **Create** to create your work pool!

Base Job Template

Defaults | Advanced

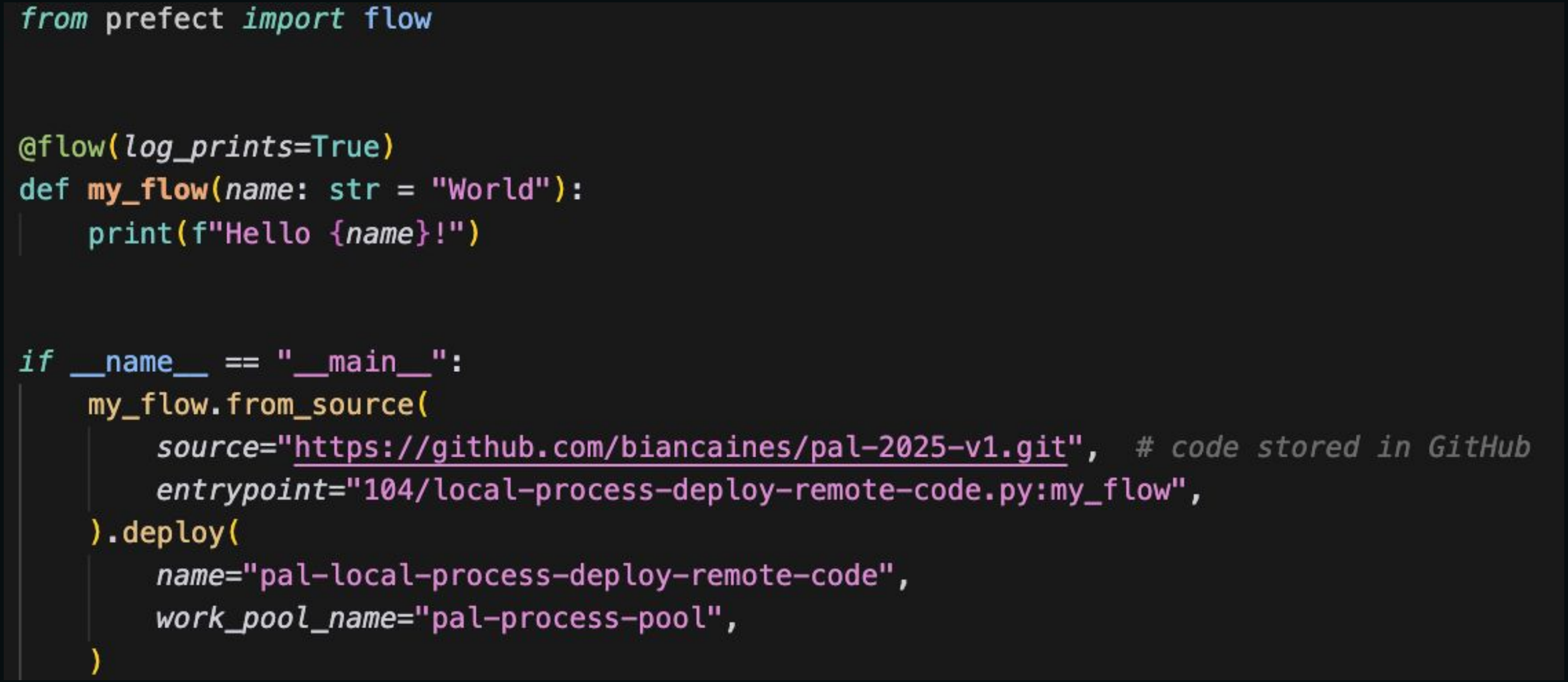
The fields below control the default values for the base job template. These values can be overridden by deployments.

**Name (Optional)**
Name given to infrastructure created by a worker.

**Environment Variables (Optional)**
Environment variables to set when starting a flow run.

**Labels (Optional)**

## Run script to create the deployment with *.deploy()*



```python
from prefect import flow


@flow(log_prints=True)
def my_flow(name: str = "World"):
    print(f"Hello {name}!")


if __name__ == "__main__":
    my_flow.from_source(
        source="https://github.com/biancaines/pal-2025-v1.git",  # code stored in GitHub
        entrypoint="104/local-process-deploy-remote-code.py:my_flow",
    ).deploy(
        name="pal-local-process-deploy-remote-code",
        work_pool_name="pal-process-pool",
    )
```

## Start a worker

- In a new terminal window
- Watches for scheduled flow runs in the work pool

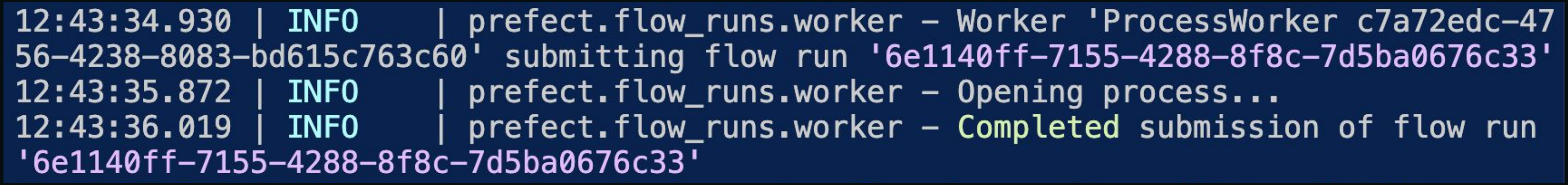
```
prefect worker start --pool 'pal-process-pool'
```

## Run the deployment using the CLI

Just like running a deployment with *.serve*

*prefect deployment run 'my-flow/pal-local-process-deploy-remote-code'*

## See flow run logs in the UI or worker’s terminal window

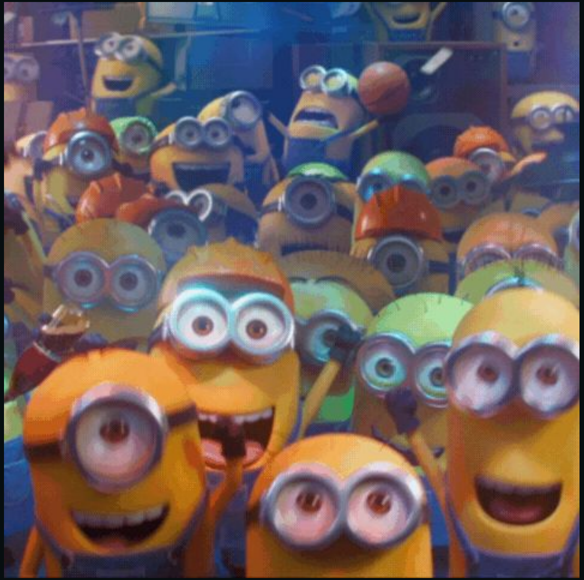


```
12:43:34.930 | INFO    | prefect.flow_runs.worker - Worker 'ProcessWorker c7a72edc-47
56-4238-8083-bd615c763c60' submitting flow run '6e1140ff-7155-4288-8f8c-7d5ba0676c33'
12:43:35.872 | INFO    | prefect.flow_runs.worker - Opening process...
12:43:36.019 | INFO    | prefect.flow_runs.worker - Completed submission of flow run
'6e1140ff-7155-4288-8f8c-7d5ba0676c33'
```

## At runtime:

1. Worker kicks off scheduled flow run
2. Pulls flow code from GitHub
3. Runs code in a local subprocess
4. Prefect monitors state
5. Subprocess exits

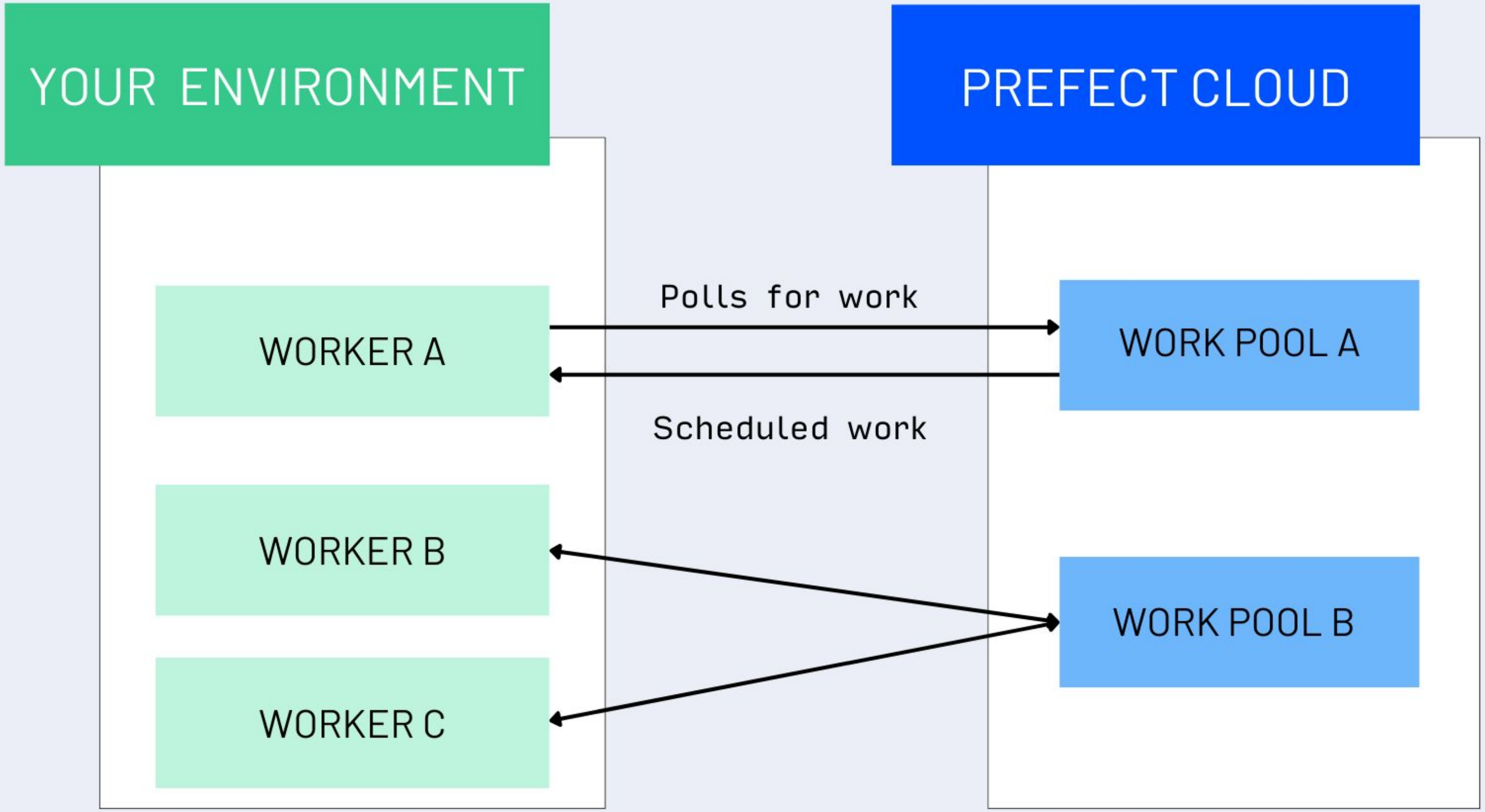
## Workers

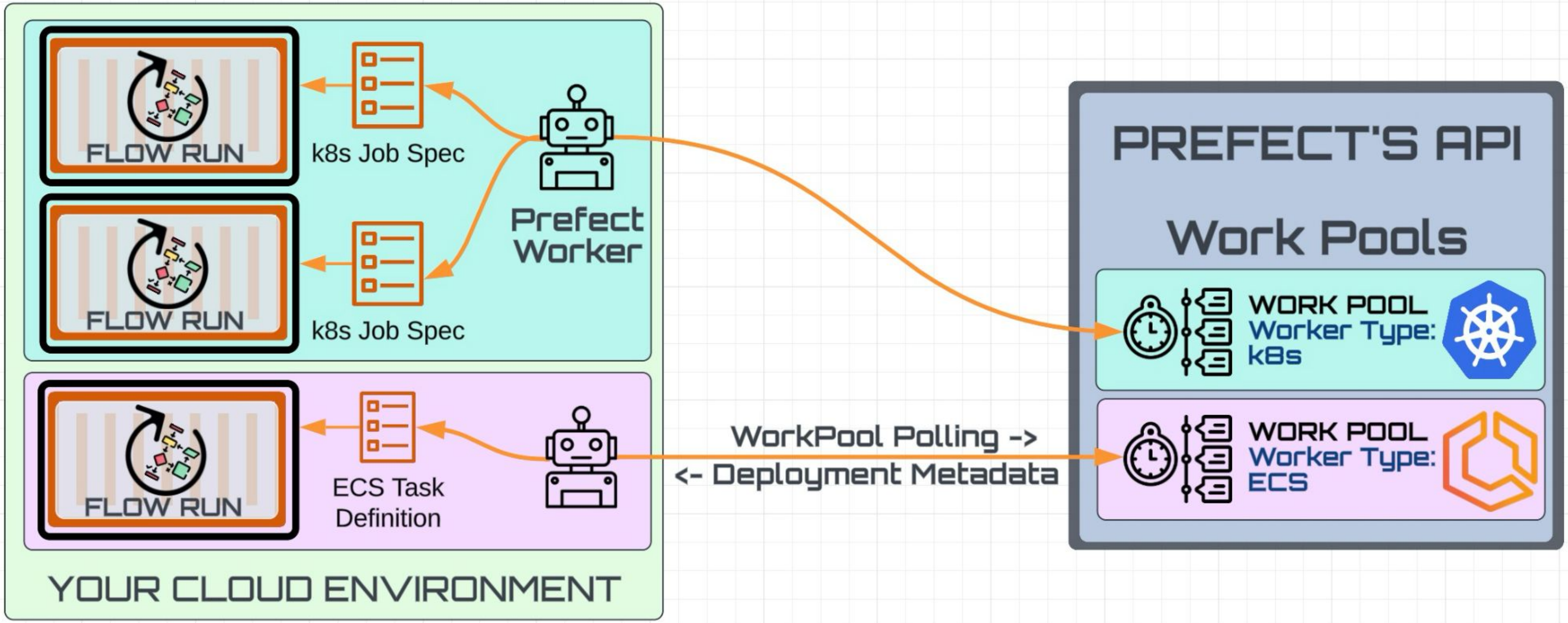


## Workers

- Long-running process on the client
- Poll for scheduled flow runs from work pools
- Must match a work pool to pick up work
- If familiar with agents (old concept), workers are like smarter, typed agents

# WORKERS & WORK POOLS




FLOW RUN
k8s Job Spec
FLOW RUN
k8s Job Spec
Prefect Worker
FLOW RUN
ECS Task Definition
YOUR CLOUD ENVIRONMENT
WorkPool Polling ->
<- Deployment Metadata
PREFECT'S API
Work Pools
WORK POOL
Worker Type: k8s
WORK POOL
Worker Type: ECS

## Example: Docker work pool & worker

## Why use Docker?

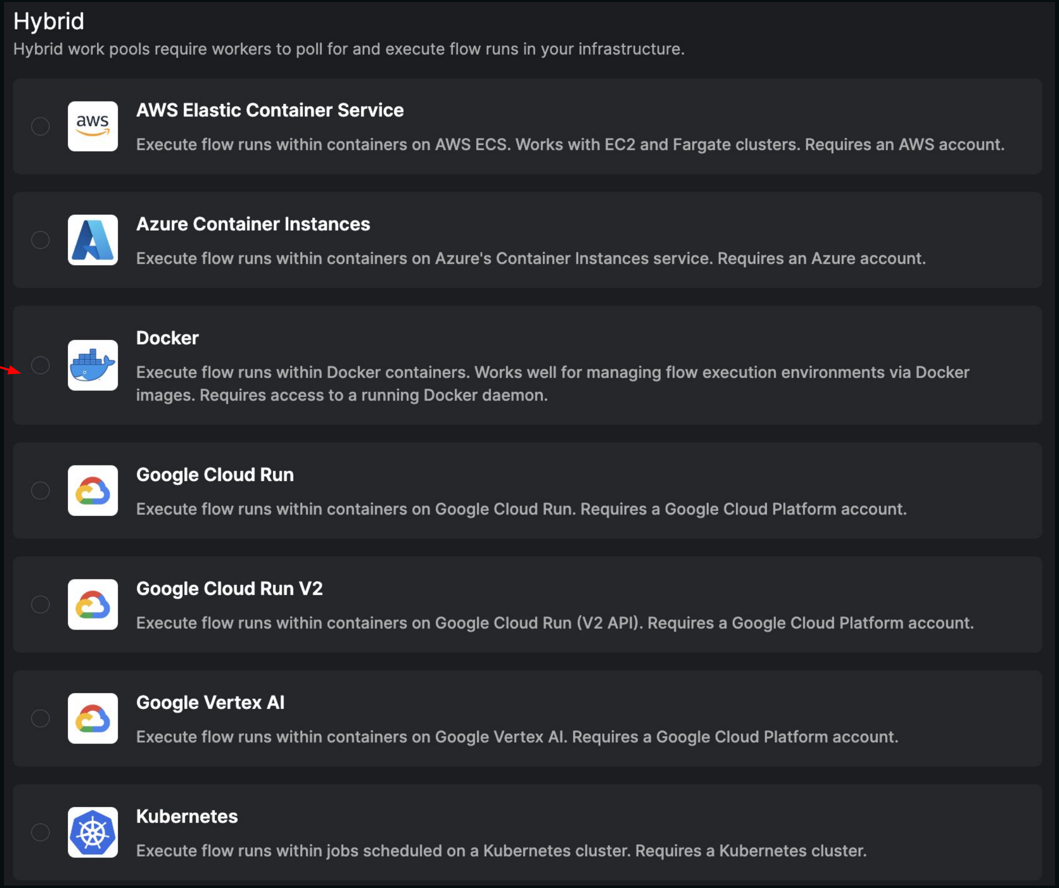
- Same operating environment everywhere
- Lighter weight than a VM
- Linux (generally)
- Portable
- Very popular
- All Prefect work pools other than Process use it

## Docker work pool

Run a flow in a Docker container

1. Start Docker on your machine
2. Create a Docker type work pool
3. Start a worker that polls the work pool
4. Create a deployment that specifies the work pool
5. Run the deployment
6. Auto spins up a Docker container & runs flow in it

# Create a Docker work pool



Hybrid

Hybrid work pools require workers to poll for and execute flow runs in your infrastructure.

**AWS Elastic Container Service**

Execute flow runs within containers on AWS ECS. Works with EC2 and Fargate clusters. Requires an AWS account.

**Azure Container Instances**

Execute flow runs within containers on Azure's Container Instances service. Requires an Azure account.

**Docker**

Execute flow runs within Docker containers. Works well for managing flow execution environments via Docker images. Requires access to a running Docker daemon.

**Google Cloud Run**

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

**Google Cloud Run V2**

Execute flow runs within containers on Google Cloud Run (V2 API). Requires a Google Cloud Platform account.

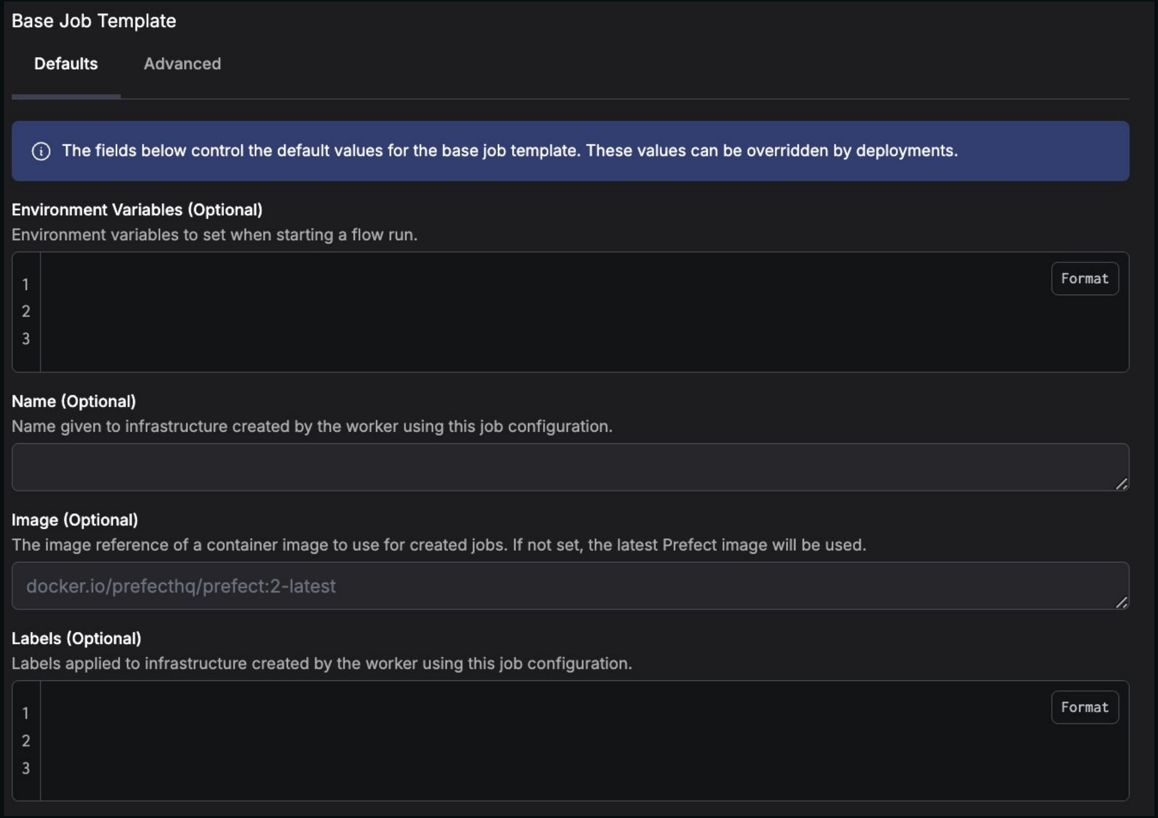
**Google Vertex AI**

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

**Kubernetes**

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

# Docker work pool - base job template



Base Job Template

Defaults    Advanced

ⓘ The fields below control the default values for the base job template. These values can be overridden by deployments.

**Environment Variables (Optional)**
Environment variables to set when starting a flow run.

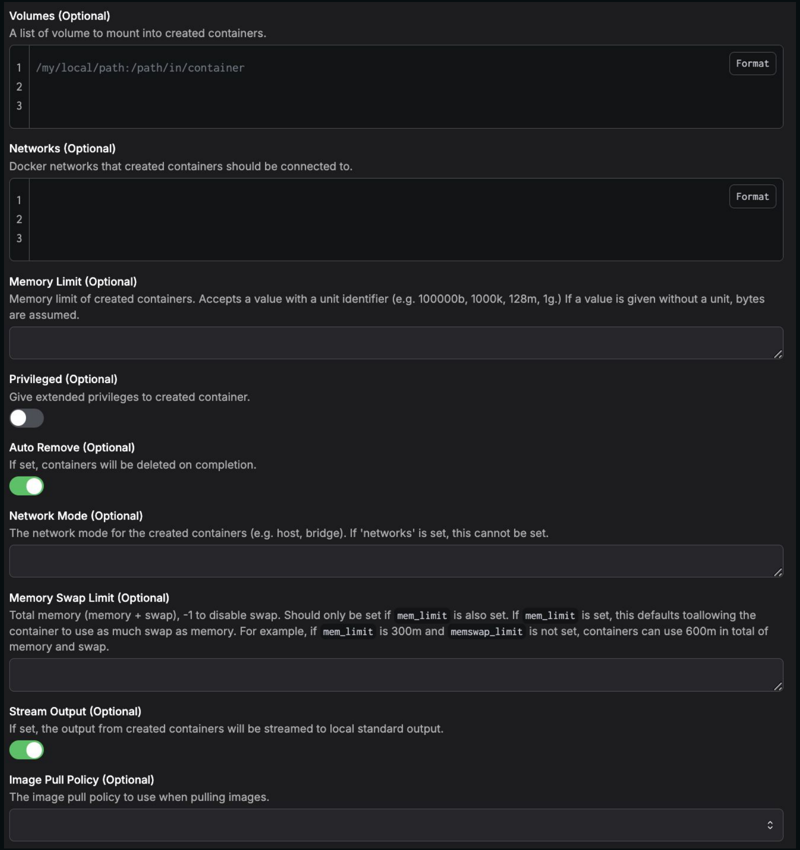
**Name (Optional)**
Name given to infrastructure created by the worker using this job configuration.

**Image (Optional)**
The image reference of a container image to use for created jobs. If not set, the latest Prefect image will be used.

docker.io/prefecthq/prefect:2-latest

**Labels (Optional)**
Labels applied to infrastructure created by the worker using this job configuration.

# Docker work pool - base job template



**Volumes (Optional)**
A list of volume to mount into created containers.

```
/my/local/path:/path/in/container
```

Format

**Networks (Optional)**
Docker networks that created containers should be connected to.

Format

**Memory Limit (Optional)**
Memory limit of created containers. Accepts a value with a unit identifier (e.g. 100000b, 1000k, 128m, 1g.) If a value is given without a unit, bytes are assumed.

**Privileged (Optional)**
Give extended privileges to created container.

**Auto Remove (Optional)**
If set, containers will be deleted on completion.

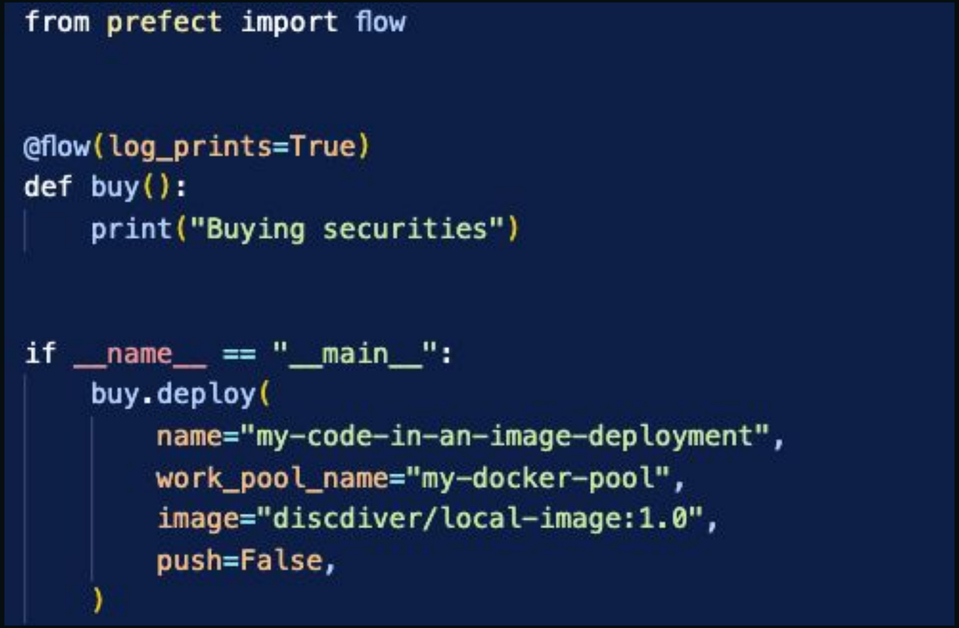
**Network Mode (Optional)**
The network mode for the created containers (e.g. host, bridge). If 'networks' is set, this cannot be set.

**Memory Swap Limit (Optional)**
Total memory (memory + swap), -1 to disable swap. Should only be set if `mem_limit` is also set. If `mem_limit` is set, this defaults toallowing the container to use as much swap as memory. For example, if `mem_limit` is 300m and `memswap_limit` is not set, containers can use 600m in total of memory and swap.

**Stream Output (Optional)**
If set, the output from created containers will be streamed to local standard output.

**Image Pull Policy (Optional)**
The image pull policy to use when pulling images.

## Package flow code into a Docker image with *.deploy()*



```python
from prefect import flow


@flow(log_prints=True)
def buy():
    print("Buying securities")


if __name__ == "__main__":
    buy.deploy(
        name="my-code-in-an-image-deployment",
        work_pool_name="my-docker-pool",
        image="discdiver/local-image:1.0",
        push=False,
    )
```

❗ *.from_source()* method not needed if baking flow code into image

## *.deploy()* method

Creates a Docker image with your flow code baked in by default!

- Specify the image name
- Specify *push=False* to not push image to registry
- Best practice: create a *requirements.txt* file with pinned package versions to install into image

## Docker type worker

Start a Docker type worker to connect to a work pool named my-*docker-pool*

*prefect worker start -p my-docker-pool*

If you want to make sure you have the packages needed:

*prefect worker start -p my-docker-pool --install-policy always*

## Docker

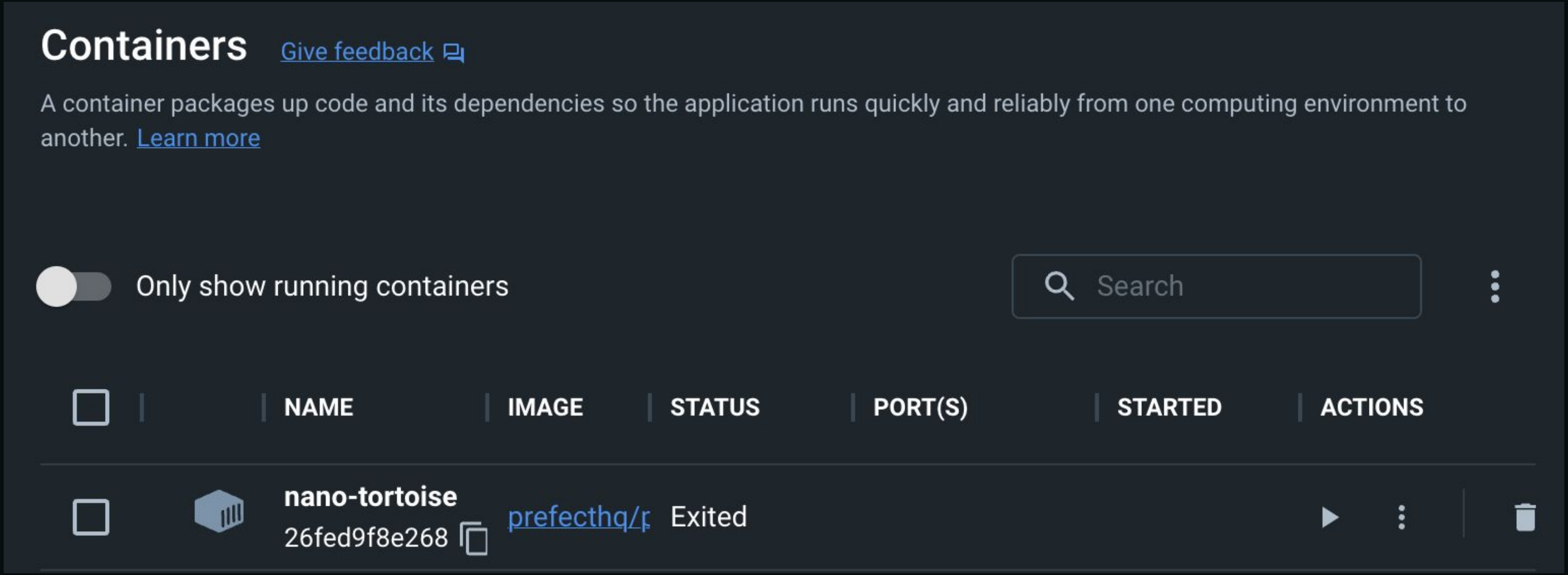
- Prefect provides base Docker images
- You can customize base image

## Docker

- Run your deployment
- Worker pulls image and spins up Docker container
- Flow code runs in Docker container and exits 🚀

## Docker

See container in Docker Desktop if running locally



Containers Give feedback

A container packages up code and its dependencies so the application runs quickly and reliably from one computing environment to another. Learn more

Only show running containers

Search

| | NAME | IMAGE | STATUS | PORT(S) | STARTED | ACTIONS |
|---|---|---|---|---|---|---|
| | nano-tortoise 26fed9f8e268 | prefecthq/p | Exited | | | |

## Docker

Reminders:

- Docker *installed* & **running**
- *prefect-docker* package installed
- Start a worker to poll for scheduled runs

## Hybrid work pool types

1. Process (local subprocess)
2. Docker
3. Serverless options such as ECS, ACI, GCR, VertexAI
4. Kubernetes

* Worker required for all

## Push work pools 📌

## Push work pools

**Serverless. No worker required.**

- AWS ECS
- Google Cloud Run
- Azure Container Instances
- Modal
- Coiled

## Push work pools

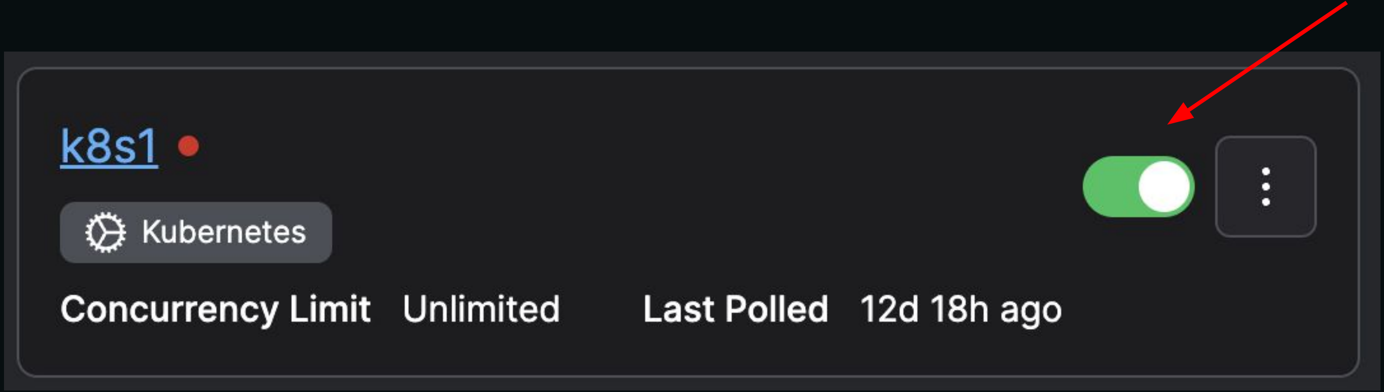
Prefect will create everything for you with *--provision-infra*

Prerequisites:

- Cloud provider account
- CLI tool installed
- Authenticated locally

```
prefect work-pool create --type modal:push --provision-infra my-modal-pool
```

## Pause **scheduled runs** for work pools from UI (or CLI)



Concurrency Limit Unlimited Last Polled 12d 18h ago

## 104 Recap

You've seen how to

- Create work-pool based deployments! 🎉
- Use the hybrid model with workers
- Bake flow code into Docker images
- Run flows on a variety of infrastructure
- Pause and resume work pools

# Lab 104

## 104 Lab

- Let's make one of our weather forecast workflows more powerful
- Create a deployment with *.deploy()* that uses a **Prefect Managed** or **Process** work pool. Reminder, **Managed** is Prefect Cloud only
- Create work pool from the UI
    - Create a deployment that references flow code stored in your own GitHub repository
        - Use your earlier fetch weather flow if you like
        - ❗ Push your code to your GitHub repo manually
    - If using a **Process** work pool start a worker to pick up scheduled flow runs
    - Run it! 🚀

## 104 Lab Extensions

**Stretch 1:** Pause and resume the work pool from the UI.

**Stretch 2:** Experiment with adjusting fields in a work pool base job template.

**Stretch 3:** If you have Docker installed:

Create a deployment where you bake your flow code into a Docker image with *.deploy()*.

- Don't push the image (or log in + push to DockerHub).

    Don't forget to:

    - Start Docker on your machine
    - Create a Docker work pool