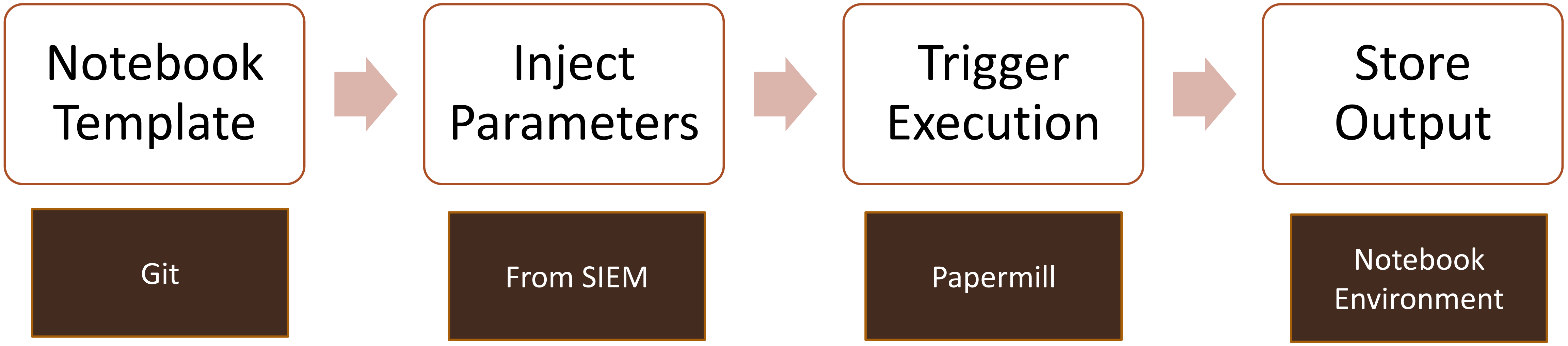
Putting it All Together

Operating Model

Automating notebooks execution allows the SOC to benefit from expert knowledge and process



Creating notebook templates

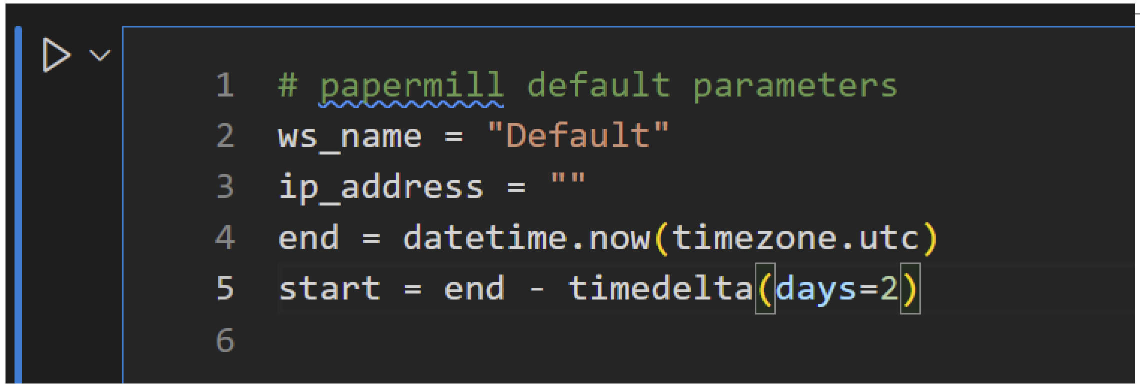
Version Control

Unattended Execution

Execution Options

- Papermill parameters
- Default execution path
- Resilient to errors
- Non-interactive

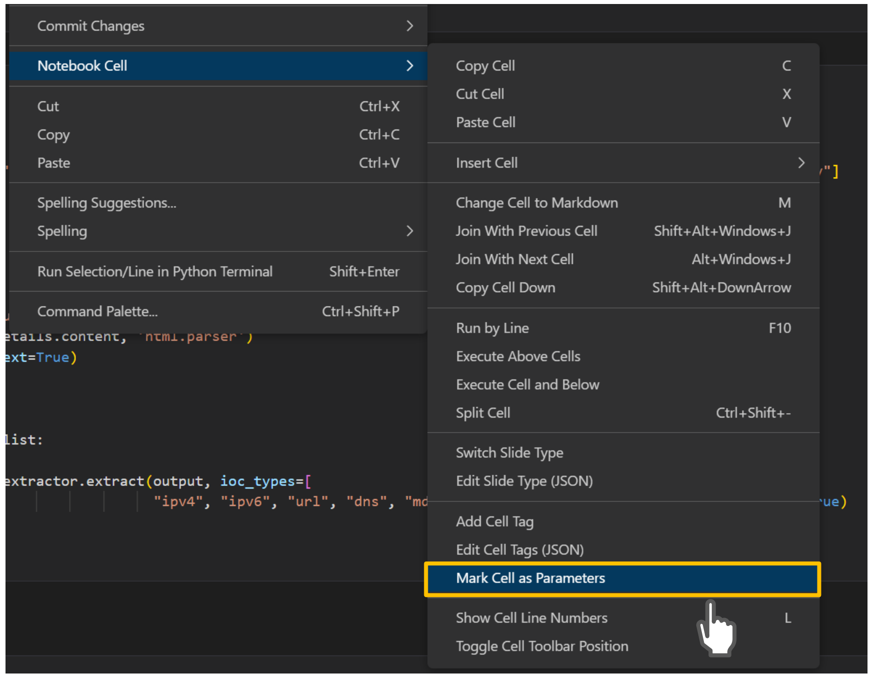
Adding Papermill Parameters



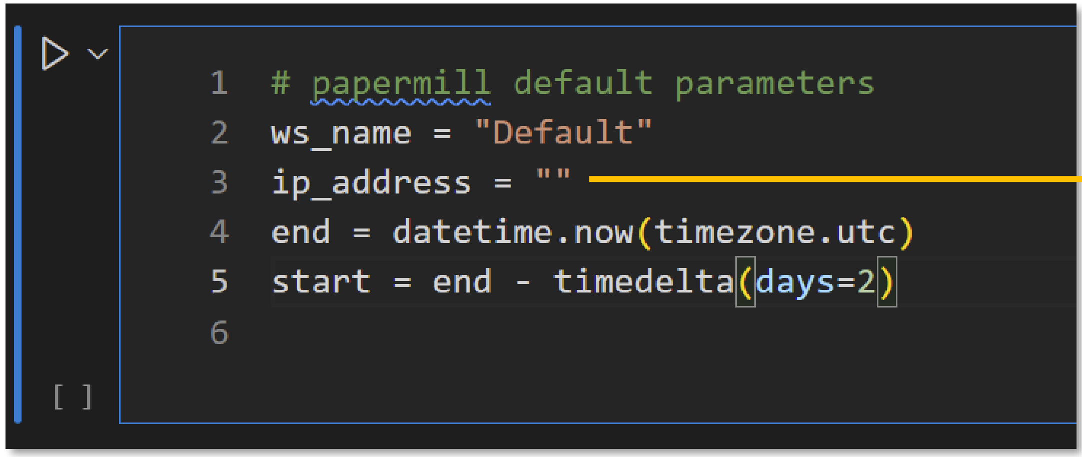
```
# papermill default parameters
ws_name = "Default"
ip_address = ""
end = datetime.now(timezone.utc)
start = end - timedelta(days=2)
```

Create template cell for parameters.
Some or all values can have defaults.

Create “parameters” cell tag.



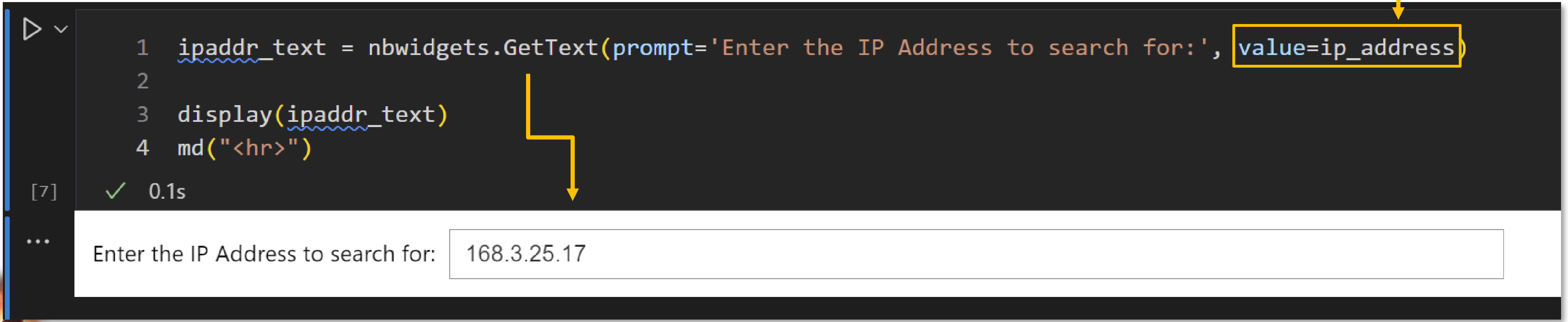
Allow interactive and automated use



```python
# papermill default parameters
ws_name = "Default"
ip_address = ""
end = datetime.now(timezone.utc)
start = end - timedelta(days=2)

```

Notebook parameter cell



```python
ipaddr_text = nbwidgets.GetText(prompt='Enter the IP Address to search for:', value=ip_address)

display(ipaddr_text)
md("<hr>")
```

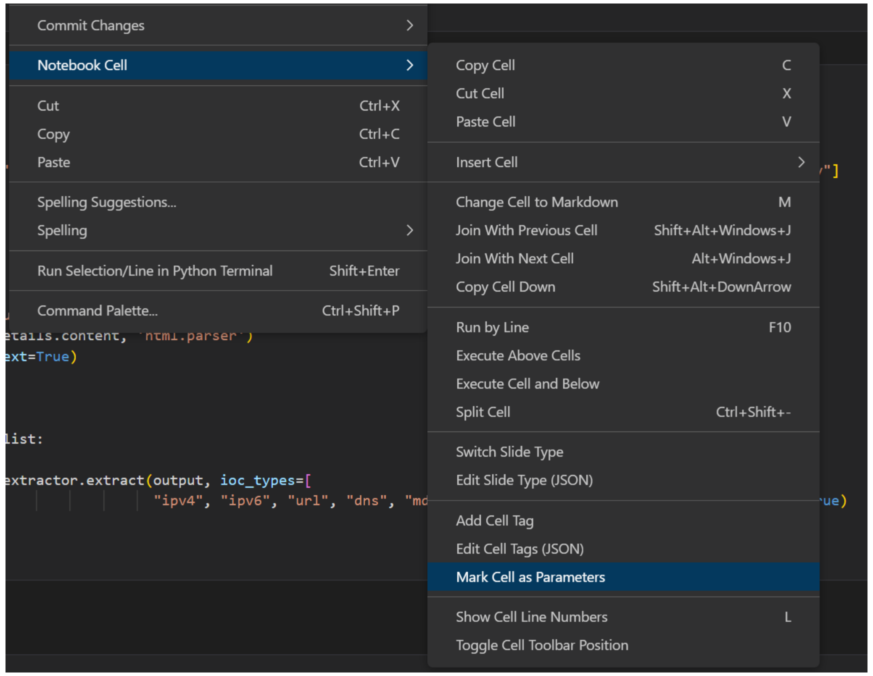
Enter the IP Address to search for: 168.3.25.17

Allow editing of parameters in non-blocking UI for interactive use

Exercise – Notebook Parameters

Open the `AutomatedNotebooks.ipynb` notebook

Right click on the cell to parameterize and select Notebook Cell > Mark Cell as Parameters



```
etails.content, 'html.parser')
ext=True)

list:

extractor.extract(output, ioc_types=[
                  "ipv4", "ipv6", "url", "dns", "md
```

Injecting parameters

1. On the command line

```
$ papermill src/ip_addr.ipynb out/output.ipynb ⏎
    -p ip_address "128.1.2.3" ⏎
    -p start "2002-07-01 13:05" ⏎
    -p end "2002-07-02 13:05"  ⏎
```

2. In a yaml file

```
ip_address: 128.1.2.3
start: 2002-07-01 13:05
end: 2002-07-02 13:05
```

```
$ papermill src/ip_addr.ipynb out/output.ipynb ⏎
    -f params.yaml
```

3. From Python

```
return pm.execute_notebook(
    input_path=input_nb,
    output_path=output_nb,
    parameters=nb_params.papermill, # Python dict
    **nb_kwargs,
)
```

Exercise – Injecting Parameters

Open up your Anaconda prompt.

Attempt to inject parameters into your AutomatedNotebook.ipynb

- `ip 115.43.212.159`

Execute the notebook with these parameters and see what output we get

Triggering execution

Scheduled - daily health checks, watch lists

On demand - investigation/analysis tasks

Event triggered - incident/alert triage

You may need **all** of these

Triggering - implementation

Use a cloud service

- Databricks, Azure Synapse, Amazon Sagemaker, etc.
- Likely need to customize for event-triggering

Roll your own

- Cron/Windows job – schedule
- File drop – on demand
- Poller – event-triggered

Build a trigger API

- HTTP endpoint
- JSON parameters

Execution - authentication and secrets

Authentication can be tricky

Data store (queries)
Services (TI)

Use a cloud service identity

Store credentials in vault (e.g. Azure Key Vault)

Avoid passing secrets/credentials as Papermill params!!!

Storing and retrieving results

Azure blob

- Cheap!

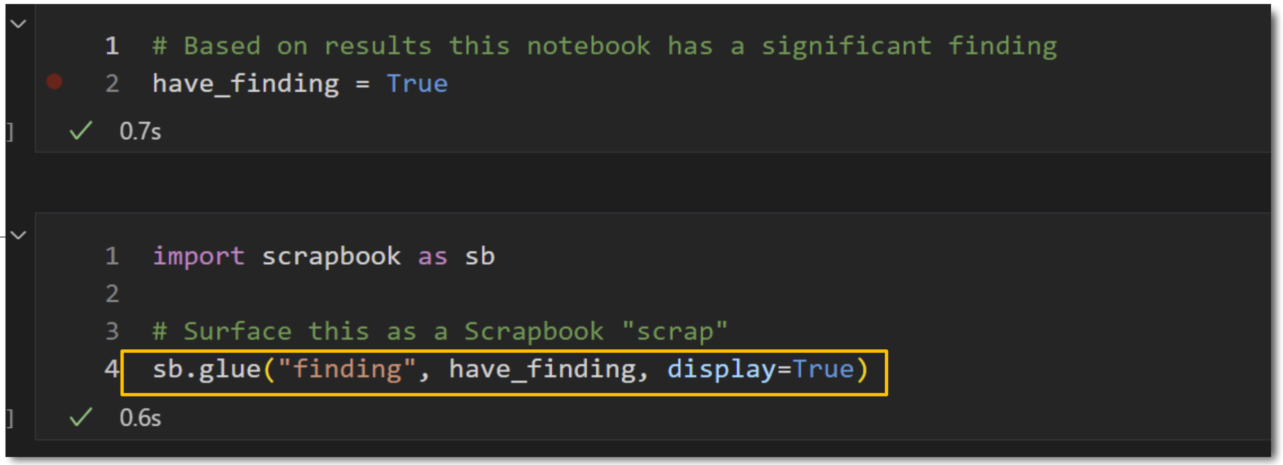
Output format

- Create output folder structure and naming scheme to organize your outputs

```
/output/2022/08/01/ip-context_124_34_13_59_{UUID}_{date}.ipynb
```

- *Papermill* can strip input code for easier reading
- Create html copies for notebooks with findings (*nbconvert*)

Storing output



```
# Based on results this notebook has a significant finding
have_finding = True
```

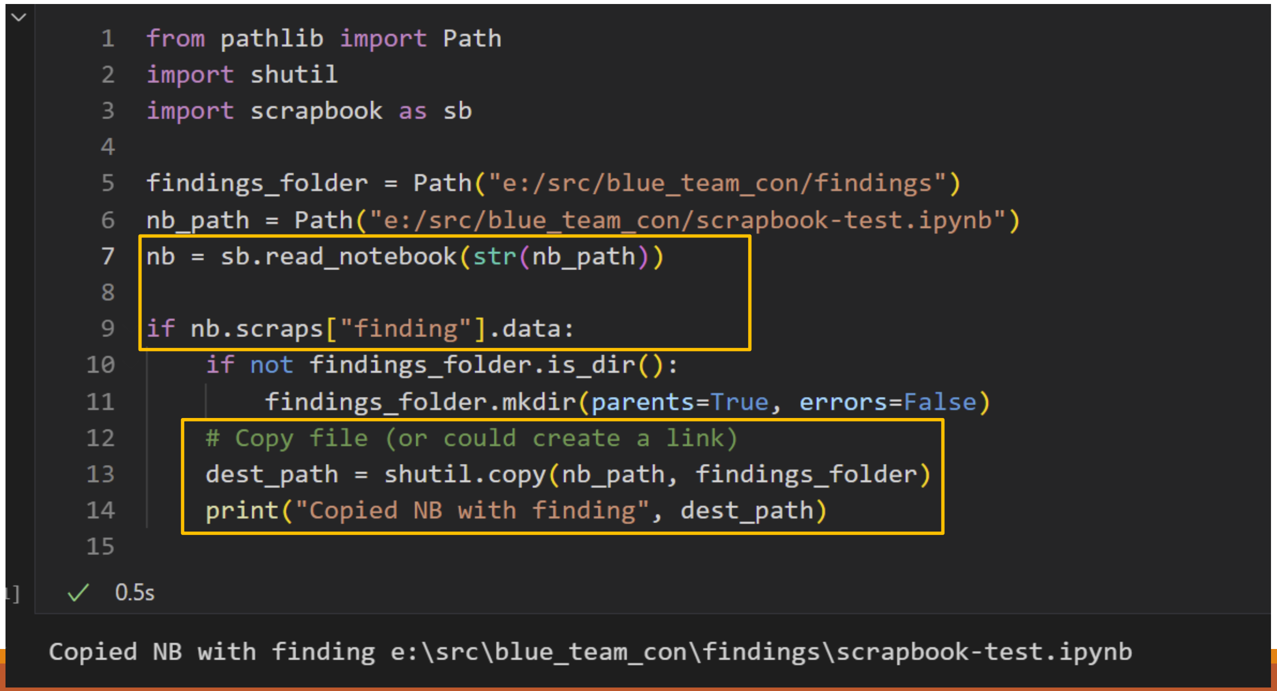
✓ 0.7s

```
import scrapbook as sb

# Surface this as a Scrapbook "scrap"
sb.glue("finding", have_finding, display=True)
```

✓ 0.6s

Use scrapbook to check for presence of the scrap



```
from pathlib import Path
import shutil
import scrapbook as sb

findings_folder = Path("e:/src/blue_team_con/findings")
nb_path = Path("e:/src/blue_team_con/scrapbook-test.ipynb")
nb = sb.read_notebook(str(nb_path))

if nb.scraps["finding"].data:
    if not findings_folder.is_dir():
        findings_folder.mkdir(parents=True, errors=False)
    # Copy file (or could create a link)
    dest_path = shutil.copy(nb_path, findings_folder)
    print("Copied NB with finding", dest_path)

```

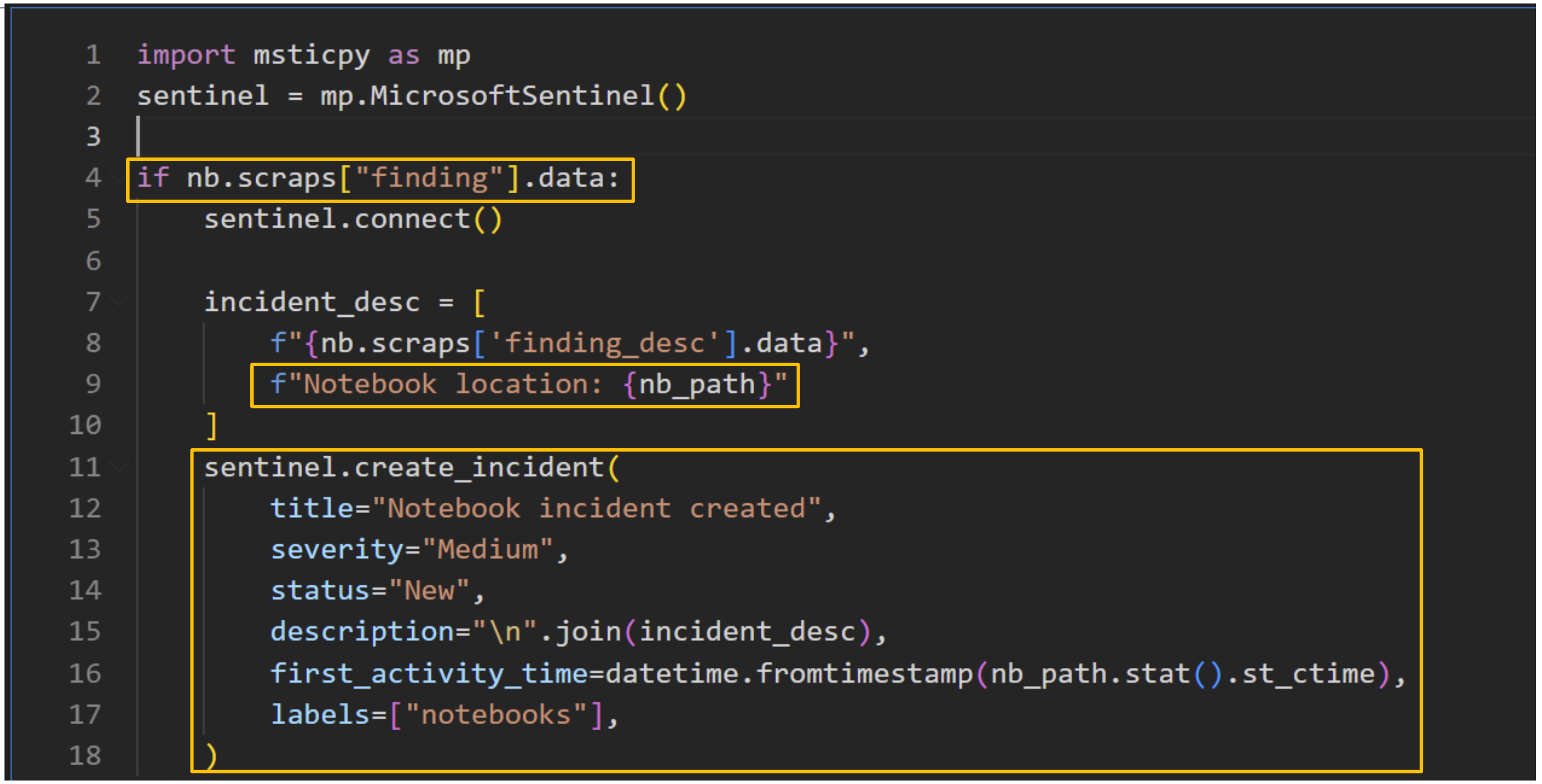
✓ 0.5s

```
Copied NB with finding e:\src\blue_team_con\findings\scrapbook-test.ipynb
```

Identifying findings: *nteract Scrapbook*

Storing output

Identifying findings: create alert/incident



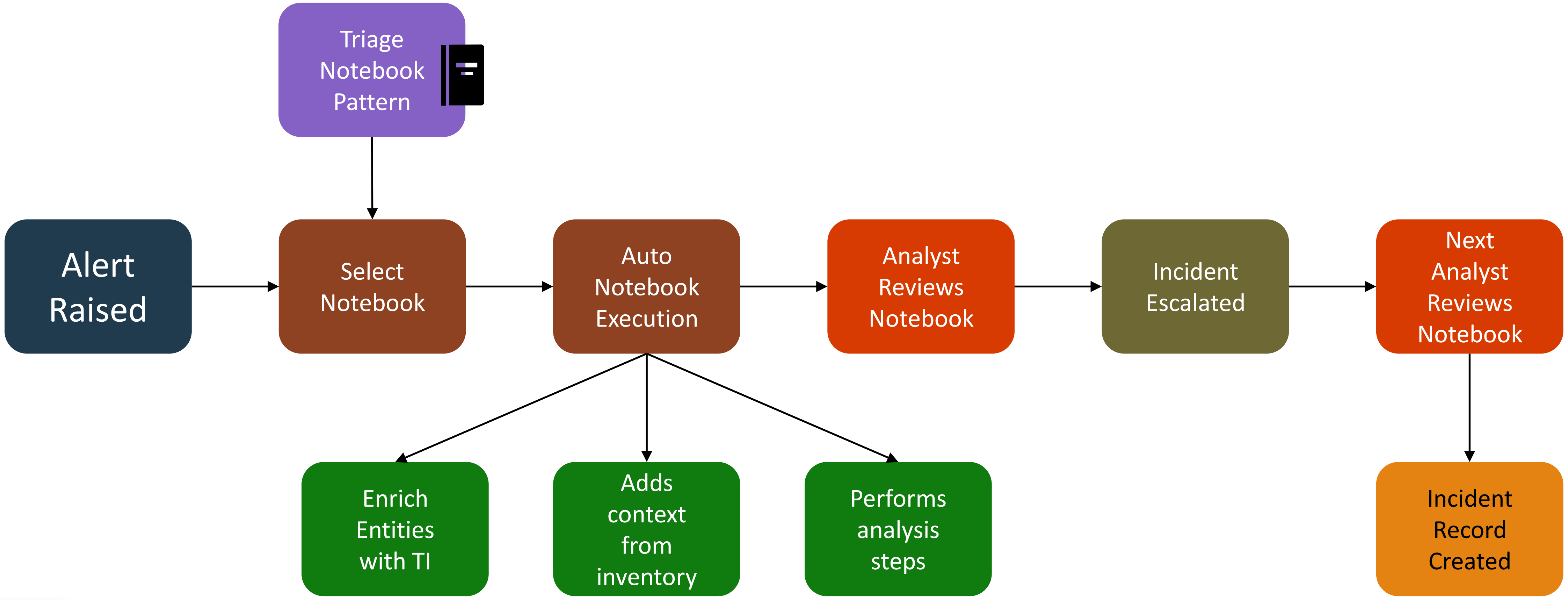
```python
import msticpy as mp
sentinel = mp.MicrosoftSentinel()

if nb.scraps["finding"].data:
    sentinel.connect()

    incident_desc = [
        f"{nb.scraps['finding_desc'].data}",
        f"Notebook location: {nb_path}"
    ]
    sentinel.create_incident(
        title="Notebook incident created",
        severity="Medium",
        status="New",
        description="\n".join(incident_desc),
        first_activity_time=datetime.fromtimestamp(nb_path.stat().st_ctime),
        labels=["notebooks"],
    )
```

Most incident management systems have equivalent mechanism

Notebooks for Alert Triage



Notebook automation examples

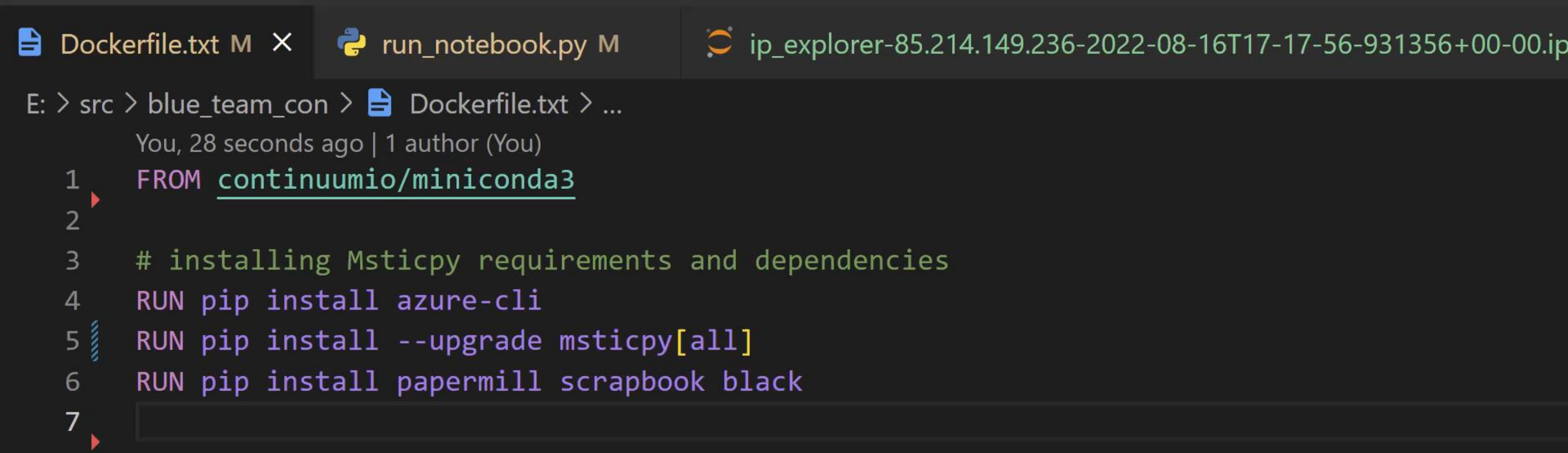
Big brother of the demo

- [Software Defined Monitoring - Using Automated Notebooks and Azure Sentinel to Improve Sec Ops]
- Create Azure VM to run notebooks triggered from incidents
- Should be adaptable to other cloud platforms

Our Demo

- Simple solution using Docker + Papermill
- Triggered by YAML parameters file
- Full source on GitHub (see refs)

Demo Time!

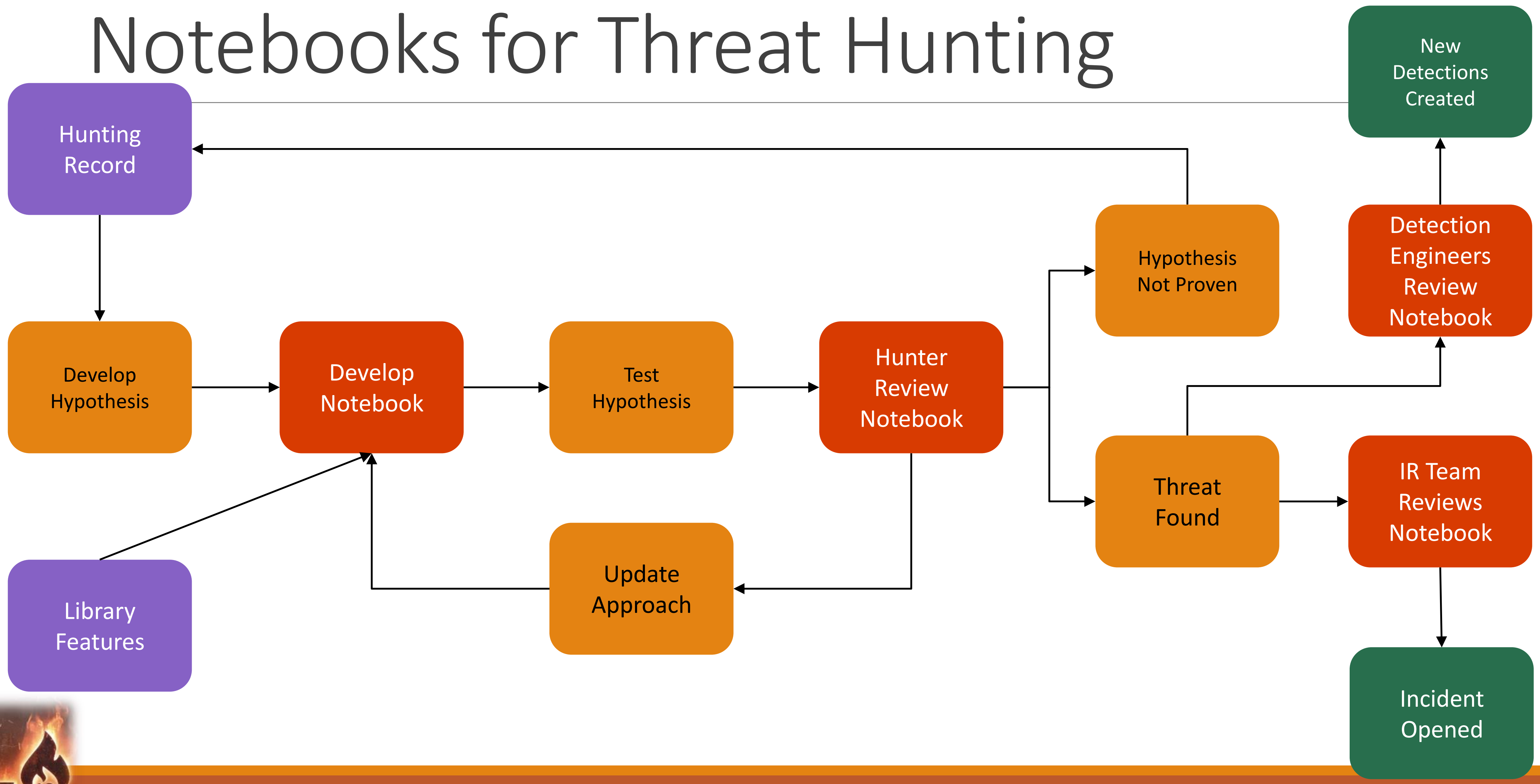

Dockerfile.txt M
run_notebook.py M
ip_explorer-85.214.149.236-2022-08-16T17-17-56-931356+00-00.ip
E: > src > blue_team_con > Dockerfile.txt > ...
You, 28 seconds ago | 1 author (You)

```
FROM continuumio/miniconda3

# installing Msticpy requirements and dependencies
RUN pip install azure-cli
RUN pip install --upgrade msticpy[all]
RUN pip install papermill scrapbook black

```

Notebooks for Threat Hunting



Threat hunting requirements

Usually **ad hoc** but may contain some automated elements

Library support is crucial – make it easy to:

- Query and retrieve information
- Create visualizations
- Repeatable analysis and data extraction/transformation

Package common tasks in parameter-driven notebooks/notebooklets

Apply the same standards as automated notebooks:

- Version control processes (for library and building-block code)
- Output naming and storage

Final Exercise - Optional

Create your own automated notebook

Take what you have learnt today and create a notebook using MSTICpy that completes some task

Parameterize the notebook

Execute notebook with injected parameters

Schedule execution for a future time