Did Something Break?

Ensuring Reliability in Bloomberg's Cloud Native Jupyter Notebook Platform

Bloomberg

Engineering

JupyterCon 2025
November 4, 2025

Rituraj Singh
Software Engineer, Cloud Native Compute Services
AI Training & Notebooks Engineering Team

TechAtBloomberg.com

© 2025 Bloomberg Finance L.P. All rights reserved.

In this talk

- AI Infrastructure at Bloomberg
- Kubernetes 101
- Jupyter Notebooks Platform
- Argo Workflows and Jupyter API
- Reliability Measures
- SLOs and SLAs

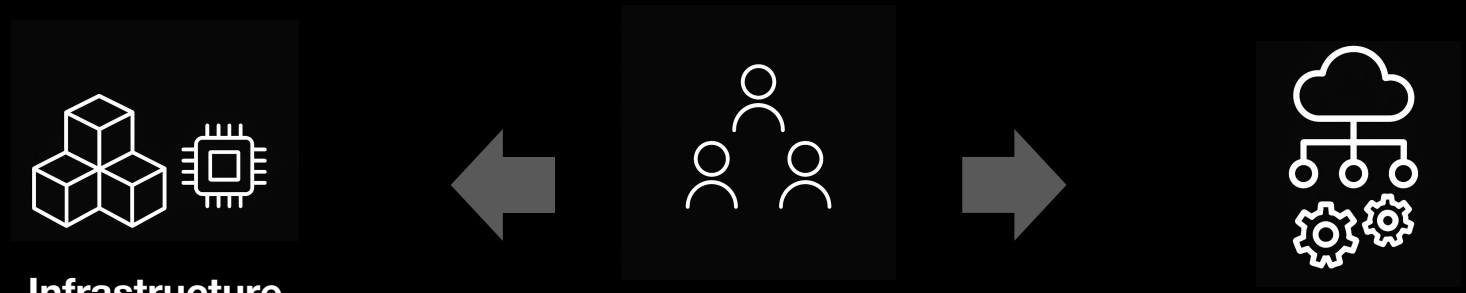
TechAtBloomberg.com

© 2025 Bloomberg Finance L.P. All rights reserved.

Bloomberg

Engineering

Bloomberg's Data Science Platform



Jupyter Notebooks

Distributed Training & HPC

Model Serving

ML Pipelines

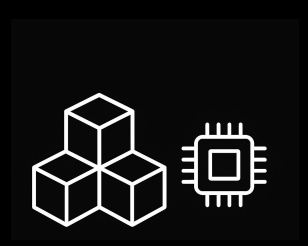
TechAtBloomberg.com

© 2025 Bloomberg Finance L.P. All rights reserved.

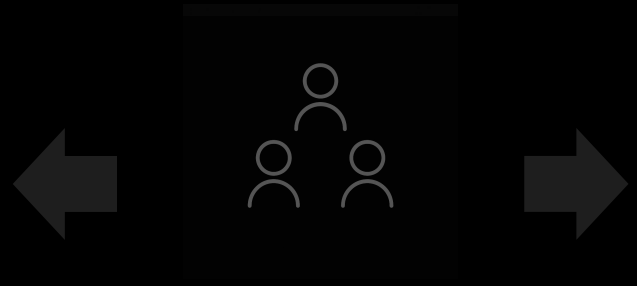
Bloomberg

Engineering

Bloomberg's Data Science Platform



Infrastructure Optimized for ML



ML Developers



ML Services & Tools



Jupyter Notebooks



Distributed Training & HPC



Model Serving



ML Pipelines



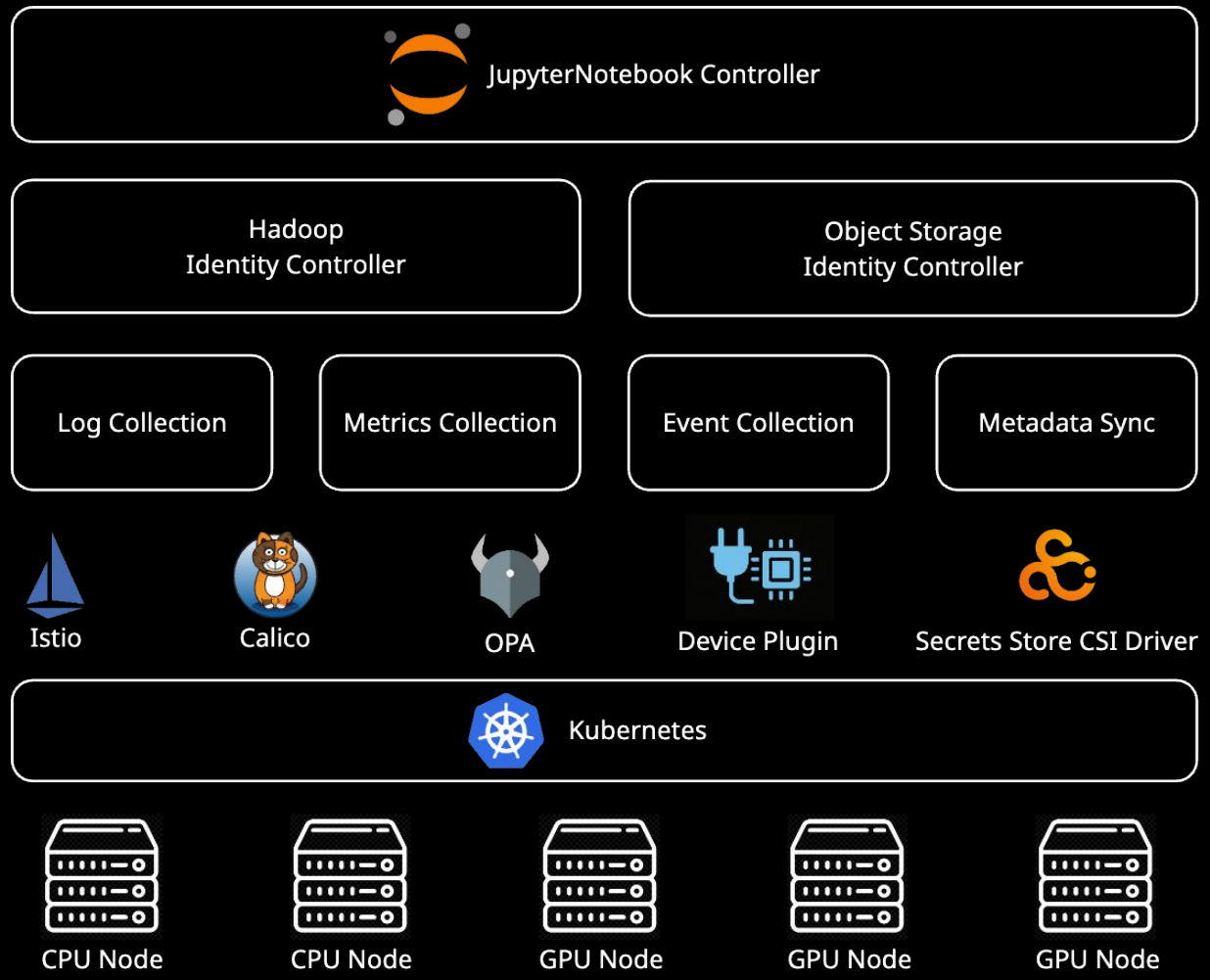
TechAtBloomberg.com

© 2025 Bloomberg Finance L.P. All rights reserved.

Bloomberg

Engineering

Jupyter Notebooks as a Service at Bloomberg



TechAtBloomberg.com

© 2025 Bloomberg Finance L.P. All rights reserved.

Bloomberg

Engineering

Kubernetes

“Kubernetes, also known as K8s, is an open source system for automating deployment, scaling, and management of containerized applications.”

Source: https://kubernetes.io/

It ensures that ML workloads and Jupyter Notebooks run seamlessly, with automatic scaling, recovery and tenant isolation.

© 2025 Bloomberg Finance L.P. All rights reserved.

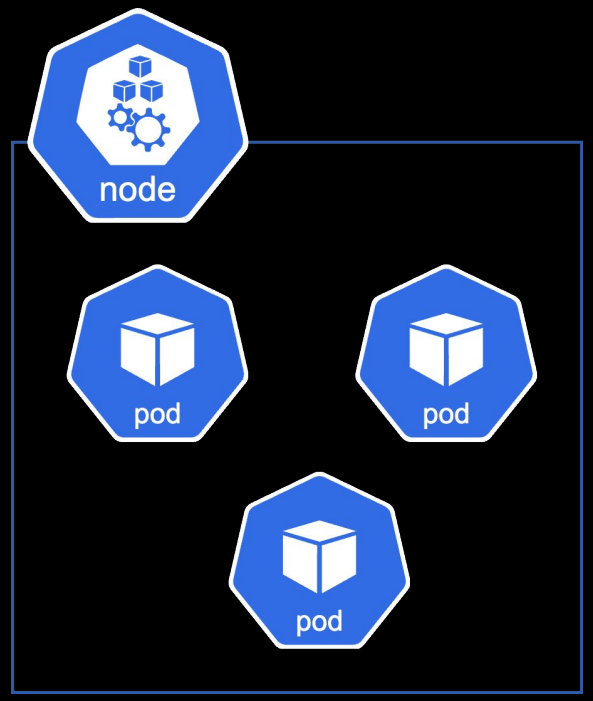
Kubernetes Building Blocks



A node is a worker machine that can run your workloads.
Each node provides compute, memory and networking resources.

© 2025 Bloomberg Finance L.P. All rights reserved.

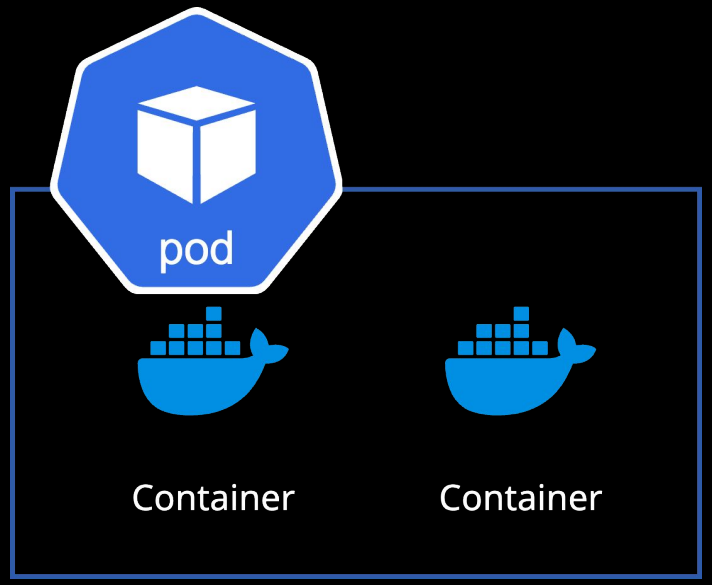
Kubernetes Building Blocks



- Pod – The smallest deployable unit in Kubernetes
- Wraps one or more containers
- Containers share:
  - The same network namespace (IP, ports)
  - The same storage volumes
- Pods are ephemeral — they can be recreated anytime

© 2025 Bloomberg Finance L.P. All rights reserved.

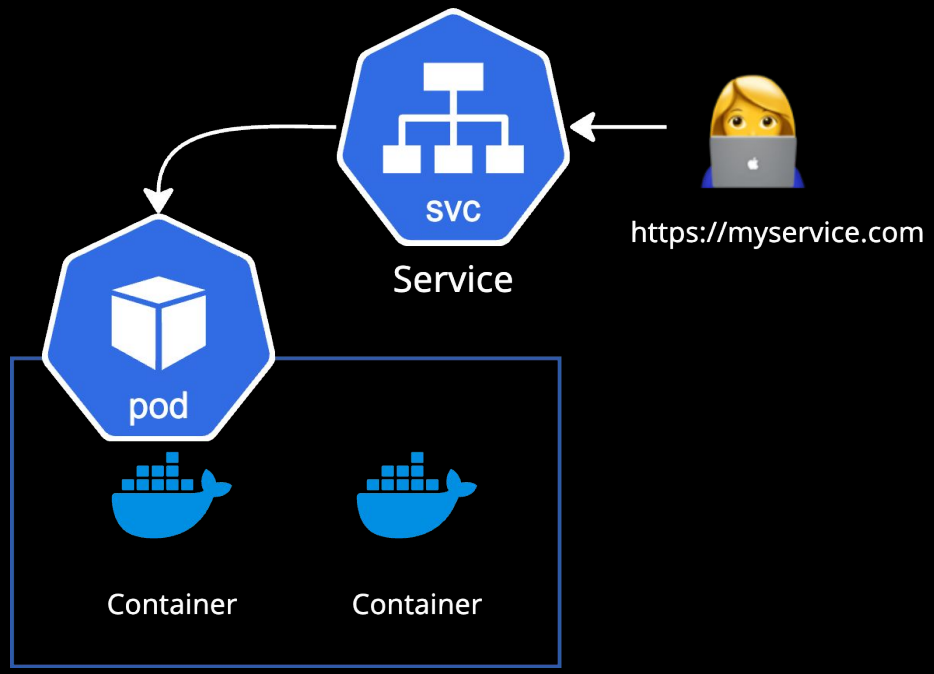
Kubernetes Building Blocks



- Containers encapsulate
  - Application code
  - Dependencies
  - Runtime configuration
- Kubernetes manages containers through Pods

© 2025 Bloomberg Finance L.P. All rights reserved.

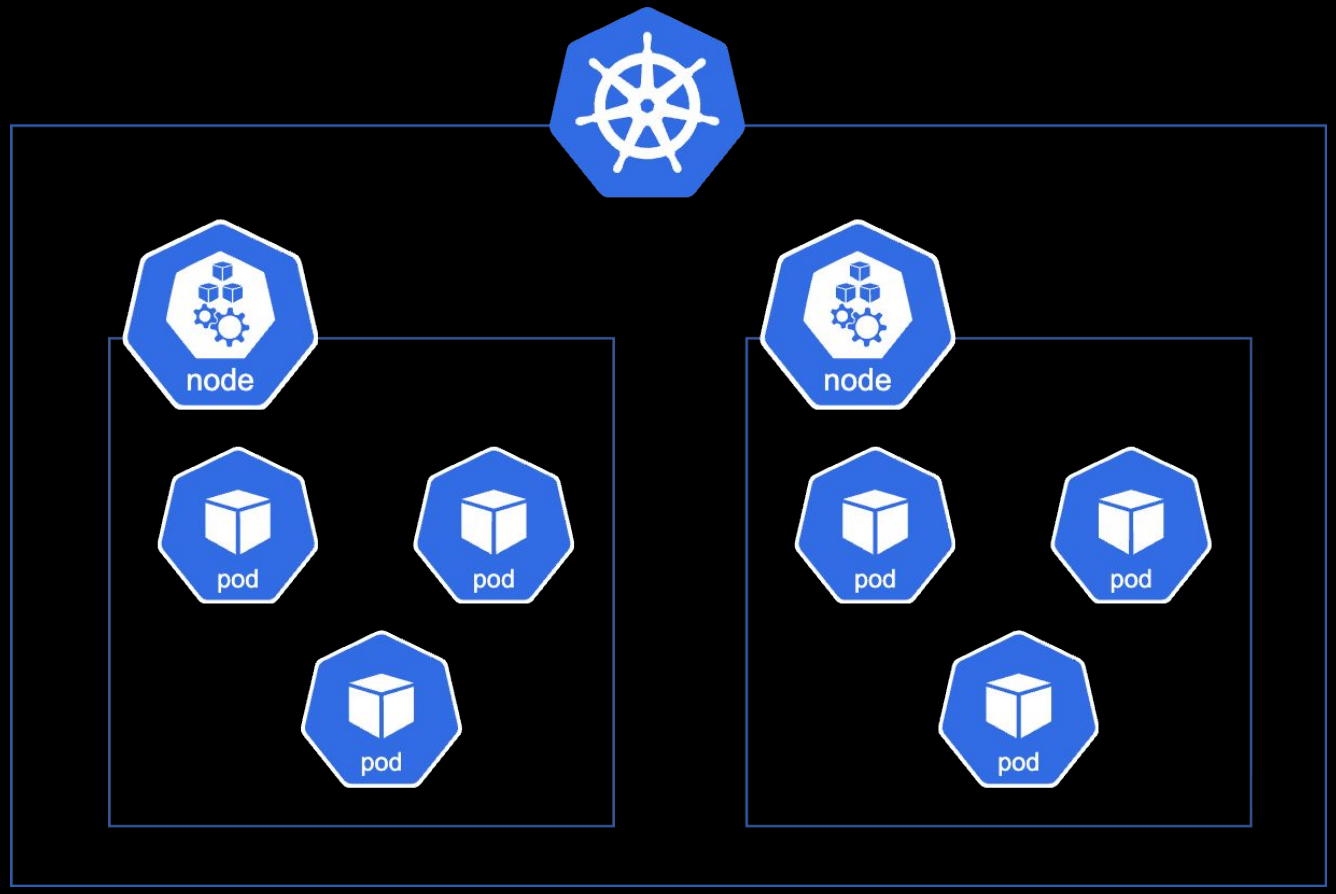
Kubernetes Building Blocks



- Provides a stable network endpoint for a set of Pods
- Abstracts away dynamic Pod IPs
- Supports different access types:
  - ClusterIP - internal access
  - NodePort - expose on node IP:port
  - LoadBalancer – external access
- Enables service discovery and load balancing

© 2025 Bloomberg Finance L.P. All rights reserved.

Kubernetes Building Blocks



A Kubernetes cluster consists of multiple nodes, each hosting pods that run your application inside containers.

© 2025 Bloomberg Finance L.P. All rights reserved.

Jupyter Notebook Platform



Request:
- 2 CPU
- 1 GPU
- 64 Gi Memory

Jupyter Notebook Spec YAML

```yaml
apiVersion: jupyter
kind: JupyterNotebook
metadata:
  name: test-notebook
  namespace: foo
spec:
  image: jupyterlab:latest
  identities:
    - s3:
        id: foo-s3
  git:
    url: 'https://github.com/foo'
  pipPackages:
    - torch
    - transformers
  resources:
    cores: 2
    gpus: 1
    memory: 64Gi
```

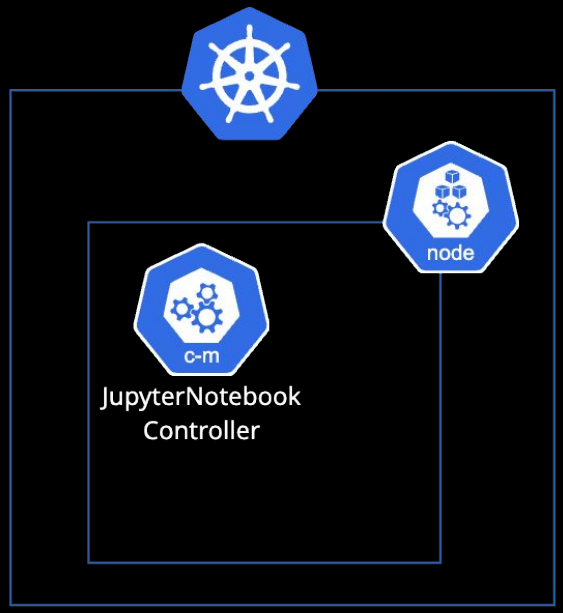

© 2025 Bloomberg Finance L.P. All rights reserved.

Jupyter Notebook Platform



Request:
2 CPU
1 GPU
64 Gi Memory

Jupyter Notebook Spec YAML



Kubernetes Notebook Cluster

A Controller parses the YAML and launches a Pod based on the spec.

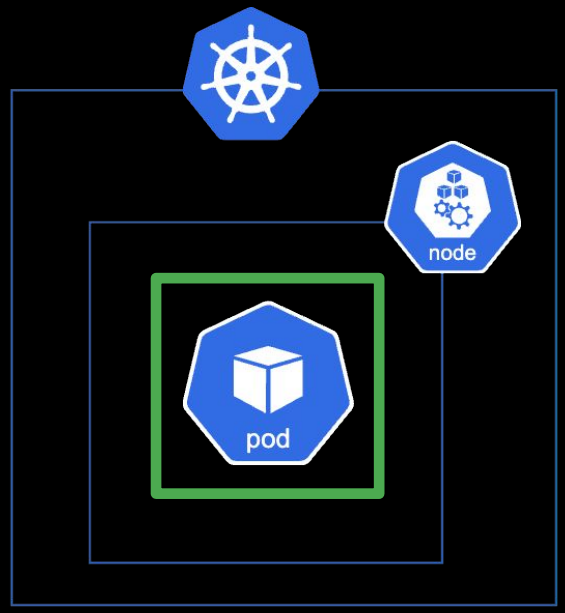
© 2025 Bloomberg Finance L.P. All rights reserved.

Jupyter Notebook Platform



Request:
- 2 CPU
- 1 GPU
- 64 Gi Memory

Jupyter Notebook Spec YAML



Kubernetes Notebook Cluster

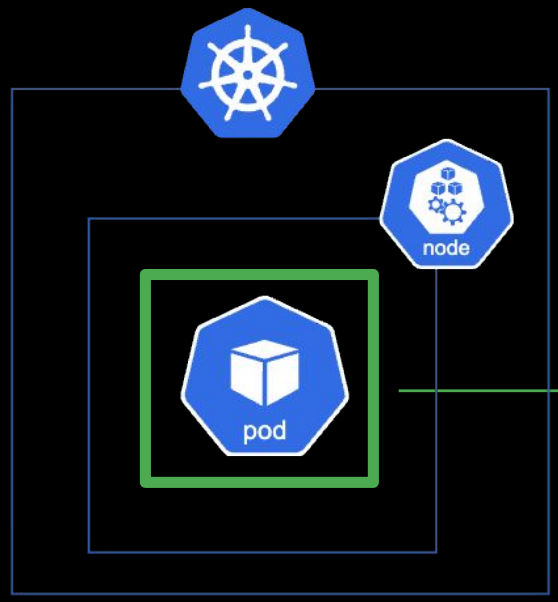
© 2025 Bloomberg Finance L.P. All rights reserved.

Jupyter Notebook Platform

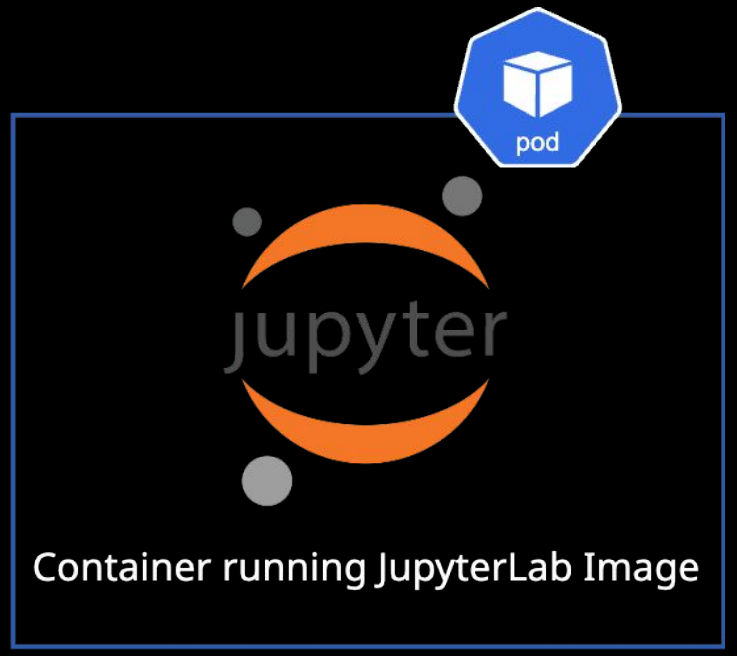


Request:
2 CPU
1 GPU
64 Gi Memory

Jupyter Notebook Spec YAML



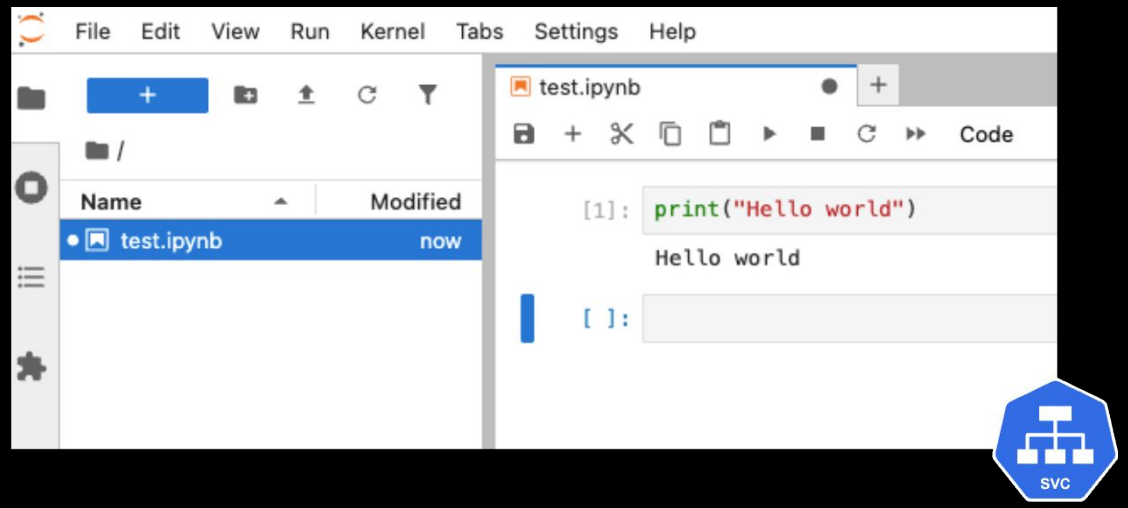
Kubernetes Notebook Cluster



© 2025 Bloomberg Finance L.P. All rights reserved.

Jupyter Notebook Platform





https://jupyternotebook.cluster.dev.com/lab

ICYMI earlier today:

“Securing & Scaling Jupyter at Bloomberg: A Kubernetes-Native Approach”
by Bloomberg’s Tucker Vento & Yifan Zhang

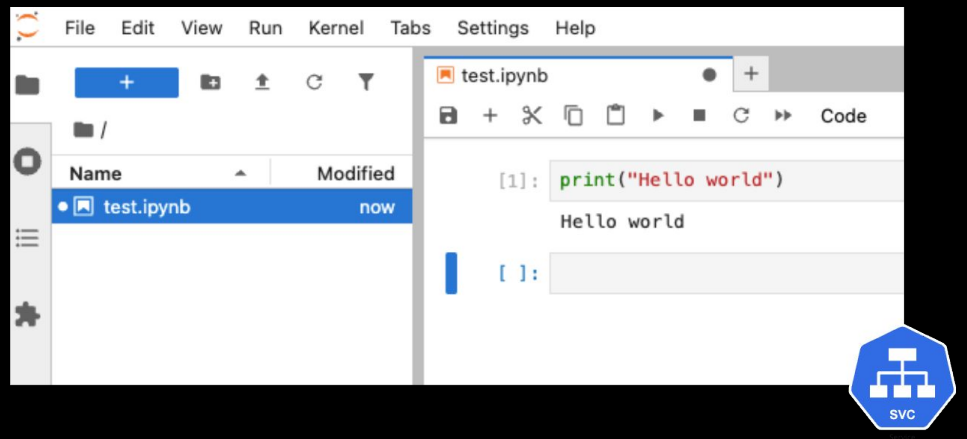
TechAtBloomberg.com

© 2025 Bloomberg Finance L.P. All rights reserved.

Bloomberg
Engineering

Behind the scenes





https://jupyternotebook.cluster.dev.com/lab

Compute upgrades

Kubernetes upgrades

Jupyter upgrades

GPU connectivity

Dependency management

Jupyter AI

Connectivity to services – models and datasets registry, LLM inference services, internal services, and more…

TechAtBloomberg.com

© 2025 Bloomberg Finance L.P. All rights reserved.

Reliability is a Feature

🌐 User Experience Monitoring

- Can users launch notebooks successfully?
- How fast are notebooks launching?
- Are notebooks accessible?

🔗 Platform & Integration Health

- Connectivity to ML tools and services
- Breakage caused by a dependency

🏗️ Infrastructure Readiness

- Is a newly built cluster ready for users?
- Workflow disruption due to Kubernetes upgrades

📊 Observability

- Key metrics and log traces
- Are we getting meaningful alerts?
- SLAs / SLOs defined and met consistently?

TechAtBloomberg.com

© 2025 Bloomberg Finance L.P. All rights reserved.

Just launch a Notebook job and check?

TechAtBloomberg.com

© 2025 Bloomberg Finance L.P. All rights reserved.

Bloomberg

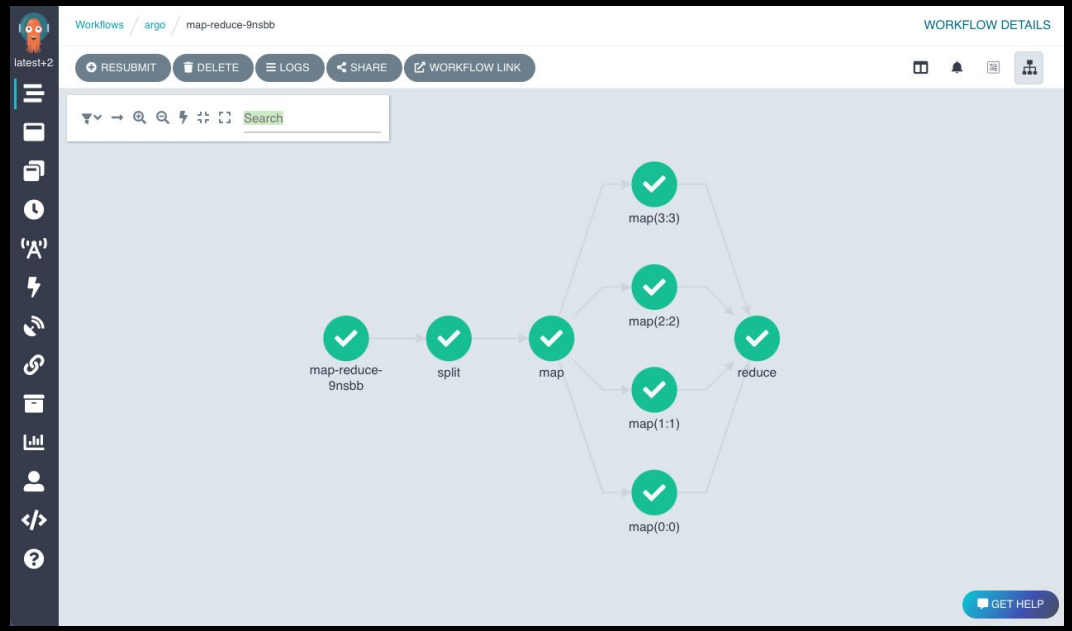
Engineering

Argo Workflows: Orchestrating Jobs on Kubernetes

Open source container-native project for orchestrating parallel jobs on Kubernetes

- Define workflow DAG
- Each step is a pod
- Configurable scheduling

Jupyter Notebooks jobs can be scheduled automatically using Argo Workflows



https://argo-workflows.readthedocs.io/en/latest/

TechAtBloomberg.com

© 2025 Bloomberg Finance L.P. All rights reserved.

Argo Workflow: Example DAG

```
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: jupyter-notebook-dag-
spec:
  entrypoint: pipeline
  templates:
  - name: pipeline
    dag:
      tasks:
      - name: run-notebook
        template: run-notebook
      - name: tester-pod
        dependencies: [run-notebook]
        template: tester-pod

  - name: run-notebook
    container:
      image: jupyter/minimal-notebook:latest
      command: [“kubectl”, “create”, “-f”, “notebook-spec.yaml”]

  - name: tester-pod
    container:
      image: jupyter-tester:latest
      command: [“kubectl”, “create”, “-f”, “tester-pod.yaml”]
```

TechAtBloomberg.com

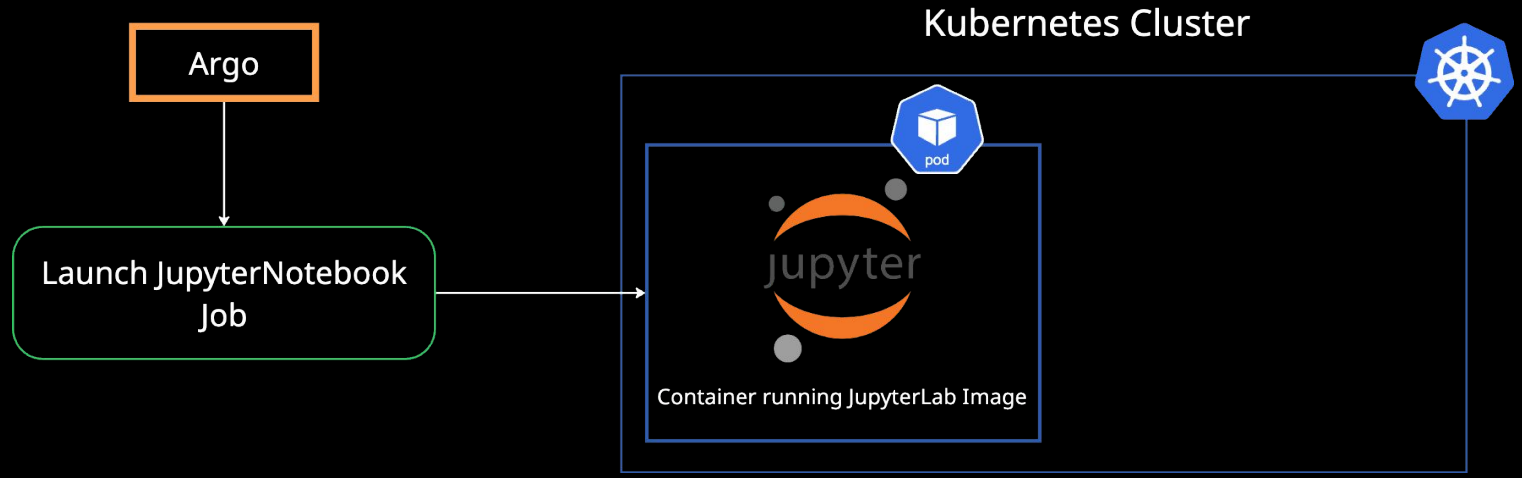
© 2025 Bloomberg Finance L.P. All rights reserved.

Argo Workflow: Example DAG

```
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: jupyter-notebook-dag-
spec:
  entrypoint: pipeline
  templates:
  - name: pipeline
    dag:
      tasks:
      - name: run-notebook
        template: run-notebook
      - name: tester-pod
        dependencies: [run-notebook]
        template: tester-pod
```

```
  - name: run-notebook
    container:
      image: jupyter/minimal-notebook:latest
      command: [“kubectl”, “create”, “-f”, “notebook-spec.yaml”]
```

```
  - name: tester-pod
    container:
      image: jupyter-tester:latest
      command: [“kubectl”, “create”, “-f”, “tester-pod.yaml”]
```



TechAtBloomberg.com

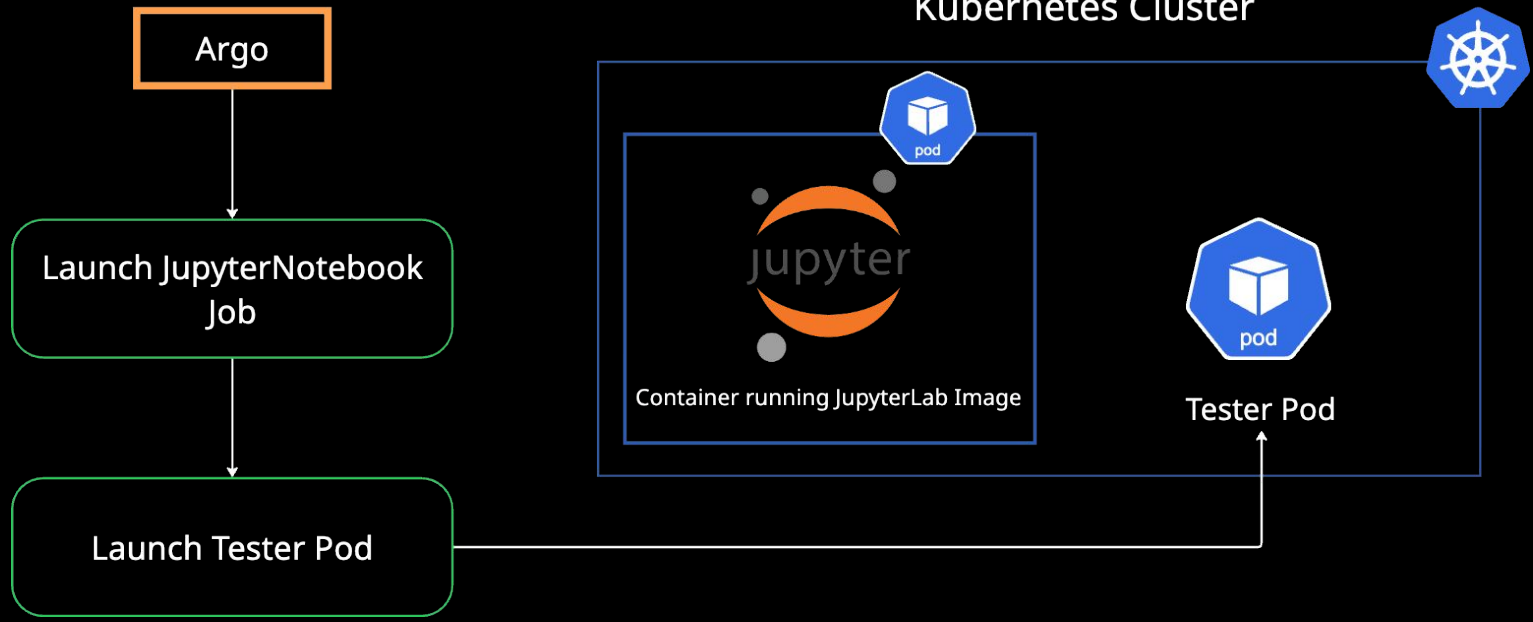
© 2025 Bloomberg Finance L.P. All rights reserved.

Argo Workflow: Example DAG

```
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: jupyter-notebook-dag-
spec:
  entrypoint: pipeline
  templates:
  - name: pipeline
    dag:
      tasks:
      - name: run-notebook
        template: run-notebook
      - name: tester-pod
        dependencies: [run-notebook]
        template: tester-pod

  - name: run-notebook
    container:
      image: jupyter/minimal-notebook:latest
      command: [“kubectl”, “create”, “-f”, “notebook-spec.yaml”]

  - name: tester-pod
    container:
      image: jupyter-tester:latest
      command: [“kubectl”, “create”, “-f”, “tester-pod.yaml”]
```



The goal of the tester pod is to interact with the Jupyter Notebook

- Create Notebooks
- Create Kernels
- Execute Code
- Evaluate Results

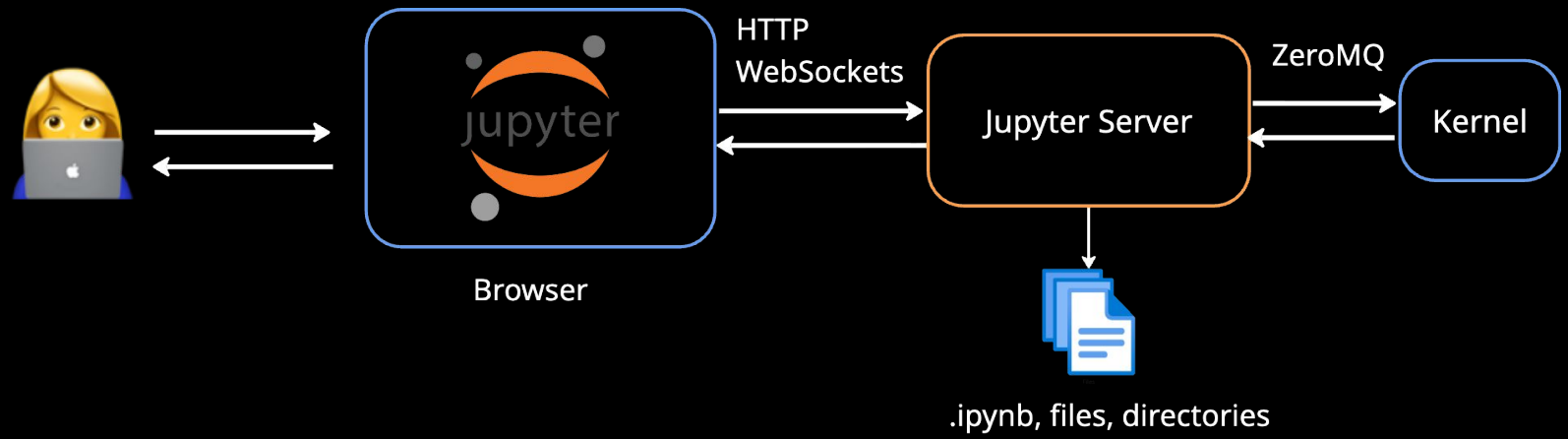
TechAtBloomberg.com

© 2025 Bloomberg Finance L.P. All rights reserved.

How does the tester pod interact with the Jupyter Notebook?

© 2025 Bloomberg Finance L.P. All rights reserved.

Behind a Notebook: Jupyter Server



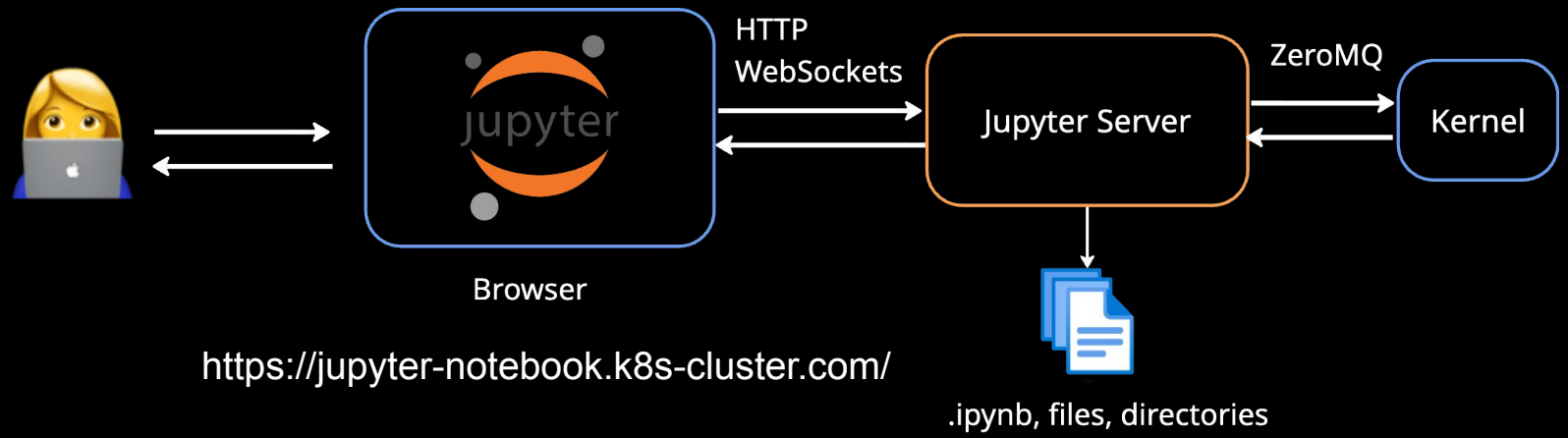
- Jupyter Server provides core services, APIs, and REST endpoints that connect the browser, kernel, and filesystem
- Communication over HTTP, WebSockets and ZeroMQ

- Creates/Deletes Notebooks
- Launches and manages kernels
- Routes code cells to the kernel, which executes the code and returns the output to the server
- File management
- Manages authentication and security tokens

TechAtBloomberg.com

© 2025 Bloomberg Finance L.P. All rights reserved.

Jupyter Server API



https://jupyter-notebook.k8s-cluster.com

- GET /api/contents/{path}
- GET /api/kernels
- GET /api/kernels/{kernel_id}
- POST /api/contents/{path}
- POST /api/kernels
- DELETE /api/contents/{path}
- DELETE /api/kernels/{kernel_id}

wss://jupyter-notebook.k8s-cluster.com/

- /api/kernels/{kernel_id}/channels
- ws.send({cell_code})

https://jupyter-server.readthedocs.io/en/latest/developers/rest-api.html

TechAtBloomberg.com

© 2025 Bloomberg Finance L.P. All rights reserved.

Tester Pod Pseudo Code

```
def create_notebook():
  requests.post(url/api/contents/{filename}, body)

def create_kernel():
  requests.post(url/api/kernels)

def kill_kernel(kernel_id):
  requests.delete(url/api/kernels/{kernel_id})

def execute_notebook_code():
  code = requests.get(url/api/contents/{notebook})
  kernel_id = create_kernel()
  ws = websocket("wss://url/api/kernels/{kernel_id}/channels")
  ws.send(code)

nb = NotebookTester(jupyter_server_url)
nb.create_notebook()
nb.execute_notebook_code()
nb.validate()
```

TechAtBloomberg.com

© 2025 Bloomberg Finance L.P. All rights reserved.

Tester Pod Pseudo Code

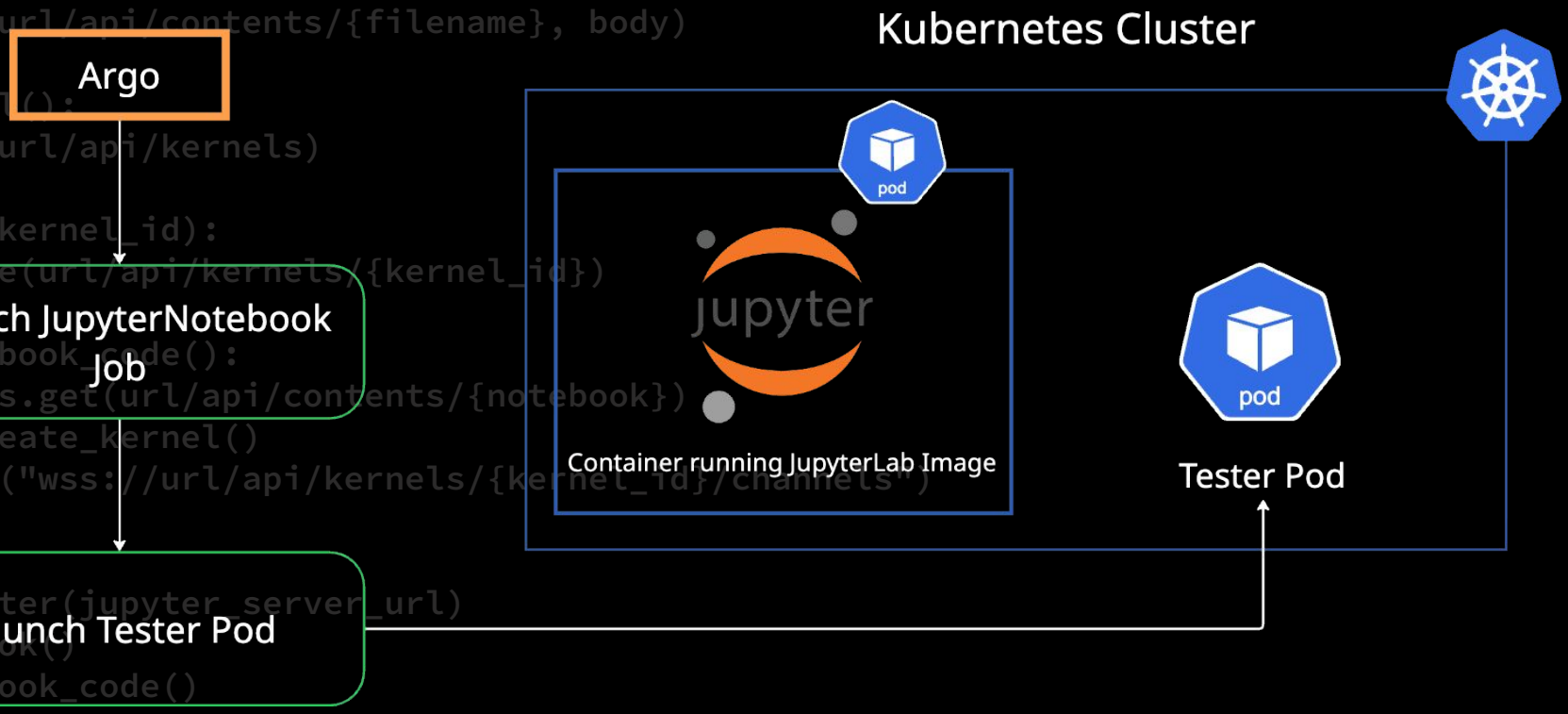
```
def create_notebook():
  requests.post(url/api/contents/{filename}, body)

def create_kernel():
  requests.post(url/api/kernels)

def kill_kernel(kernel_id):
  requests.delete(url/api/kernels/{kernel_id})

def execute_notebook_code():
  code = requests.get(url/api/contents/{notebook})
  kernel_id = create_kernel()
  ws = websocket("wss://url/api/kernels/{kernel_id}/channels")
  ws.send(code)

nb = NotebookTester(jupyter_server_url)
nb.create_notebook()
nb.execute_notebook_code()
nb.validate()
```



TechAtBloomberg.com

© 2025 Bloomberg Finance L.P. All rights reserved.

Tester Pod Pseudo Code

```
def create_notebook():
  requests.post(url/api/contents/{filename}, body)

def create_kernel():
  requests.post(url/api/kernels)

def kill_kernel(kernel_id):
  requests.delete(url/api/kernels/{kernel_id})

def execute_notebook_code():
  code = requests.get(url/api/contents/{notebook})
  kernel_id = create_kernel()
  ws = websocket("wss://url/api/kernels/{kernel_id}/channels")
  ws.send(code)

nb = NotebookTester(jupyter_server_url)
nb.create_notebook()
nb.execute_notebook_code()
nb.validate()
```

Tester Pod Container has 3 artifacts

- .ipynb files that need to be executed
- script.py for calling Jupyter API
- expected_assertions.json for validation



TechAtBloomberg.com

© 2025 Bloomberg Finance L.P. All rights reserved.

Tester Pod Pseudo Code

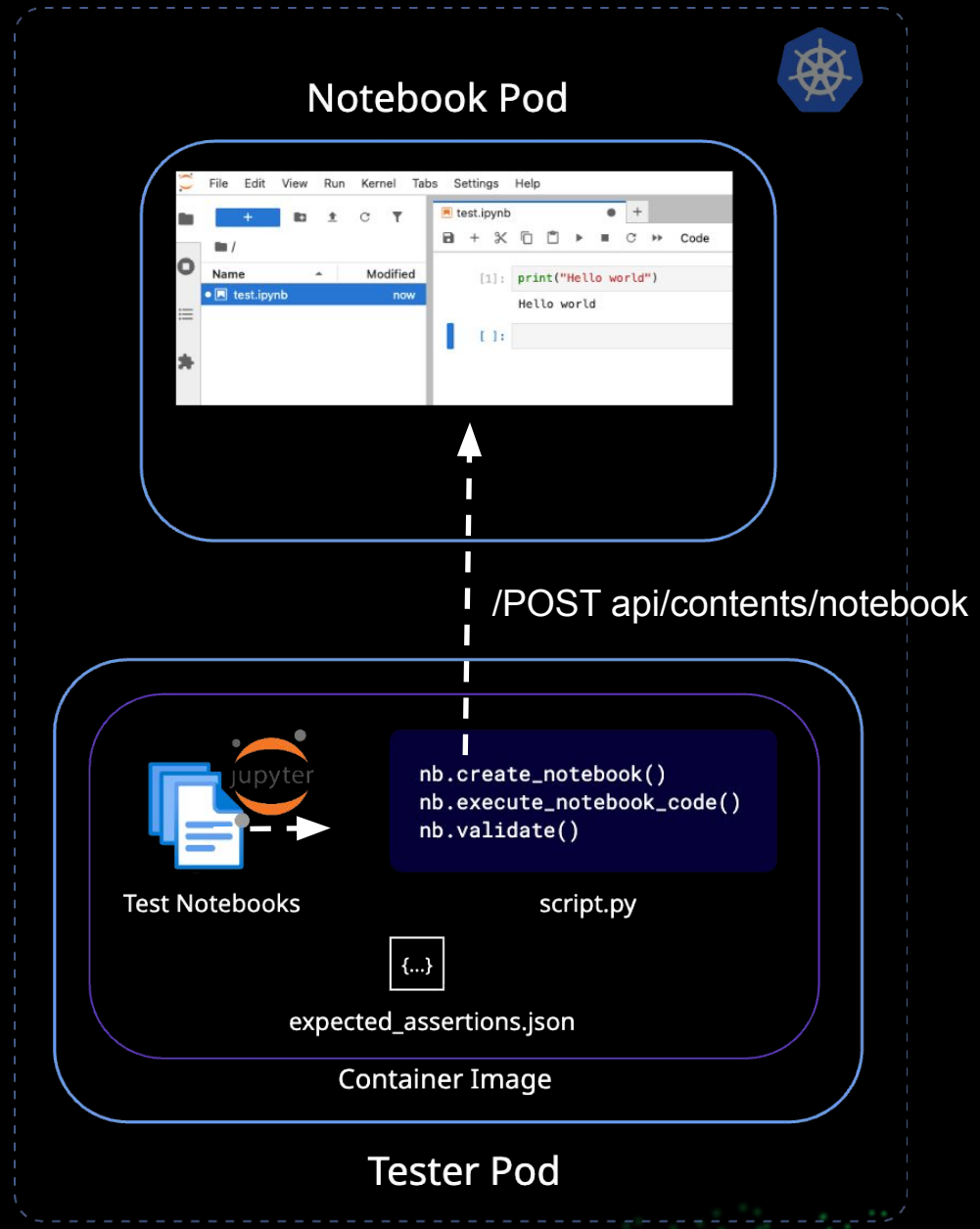
```
def create_notebook():
  requests.post(url/api/contents/{filename}, body)

def create_kernel():
  requests.post(url/api/kernels)

def kill_kernel(kernel_id):
  requests.delete(url/api/kernels/{kernel_id})

def execute_notebook_code():
  code = requests.get(url/api/contents/{notebook})
  kernel_id = create_kernel()
  ws = websocket("wss://url/api/kernels/{kernel_id}/channels")
  ws.send(code)

nb = NotebookTester(jupyter_server_url)
nb.create_notebook()
nb.execute_notebook_code()
nb.validate()
```



TechAtBloomberg.com

© 2025 Bloomberg Finance L.P. All rights reserved.

Tester Pod Pseudo Code

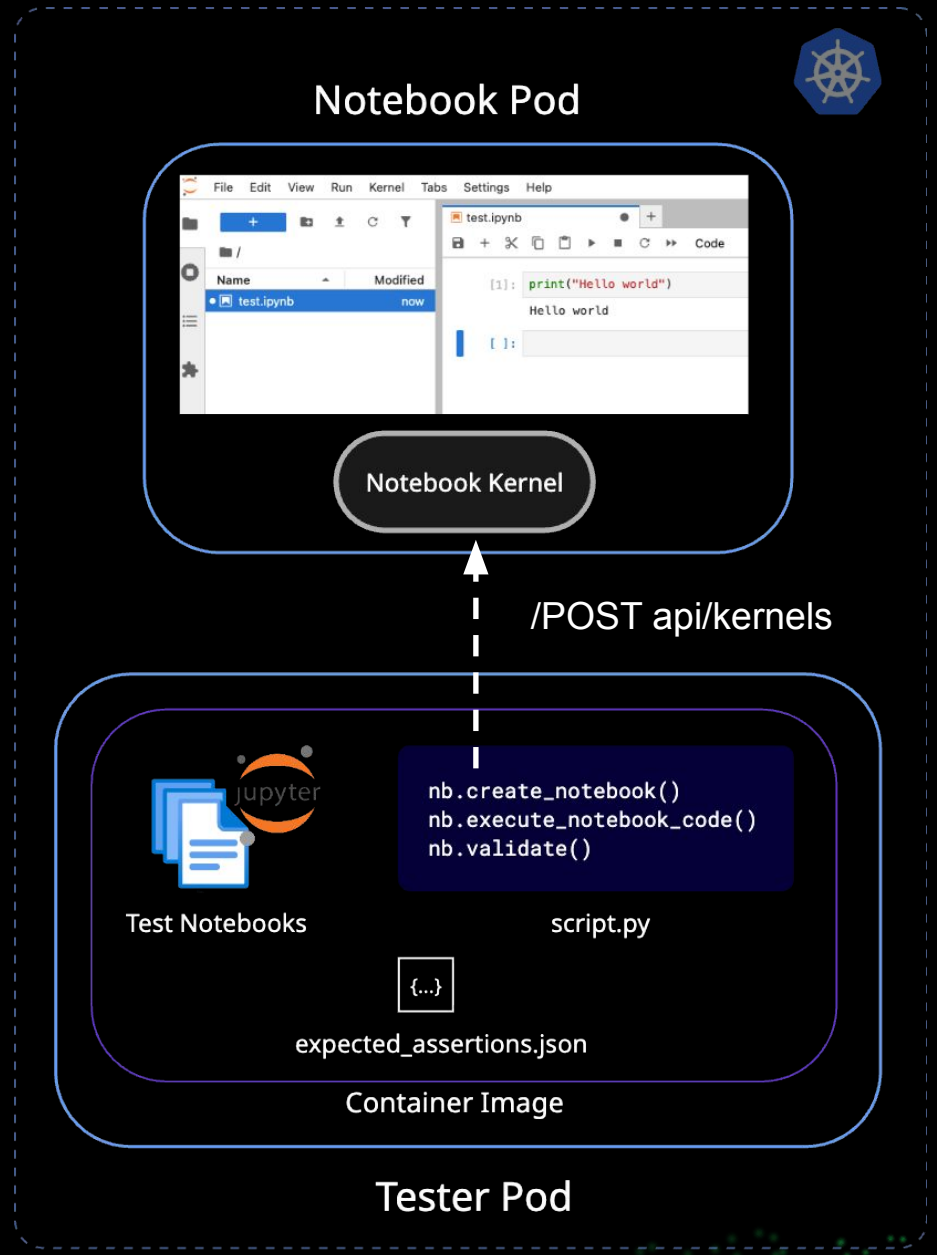
```
def create_notebook():
  requests.post(url/api/contents/{filename}, body)

def create_kernel():
  requests.post(url/api/kernels)

def kill_kernel(kernel_id):
  requests.delete(url/api/kernels/{kernel_id})

def execute_notebook_code():
  code = requests.get(url/api/contents/{notebook})
  kernel_id = create_kernel()
  ws = websocket("wss://url/api/kernels/{kernel_id}/channels")
  ws.send(code)

nb = NotebookTester(jupyter_server_url)
nb.create_notebook()
nb.execute_notebook_code()
nb.validate()
```



TechAtBloomberg.com

© 2025 Bloomberg Finance L.P. All rights reserved.

Tester Pod Pseudo Code

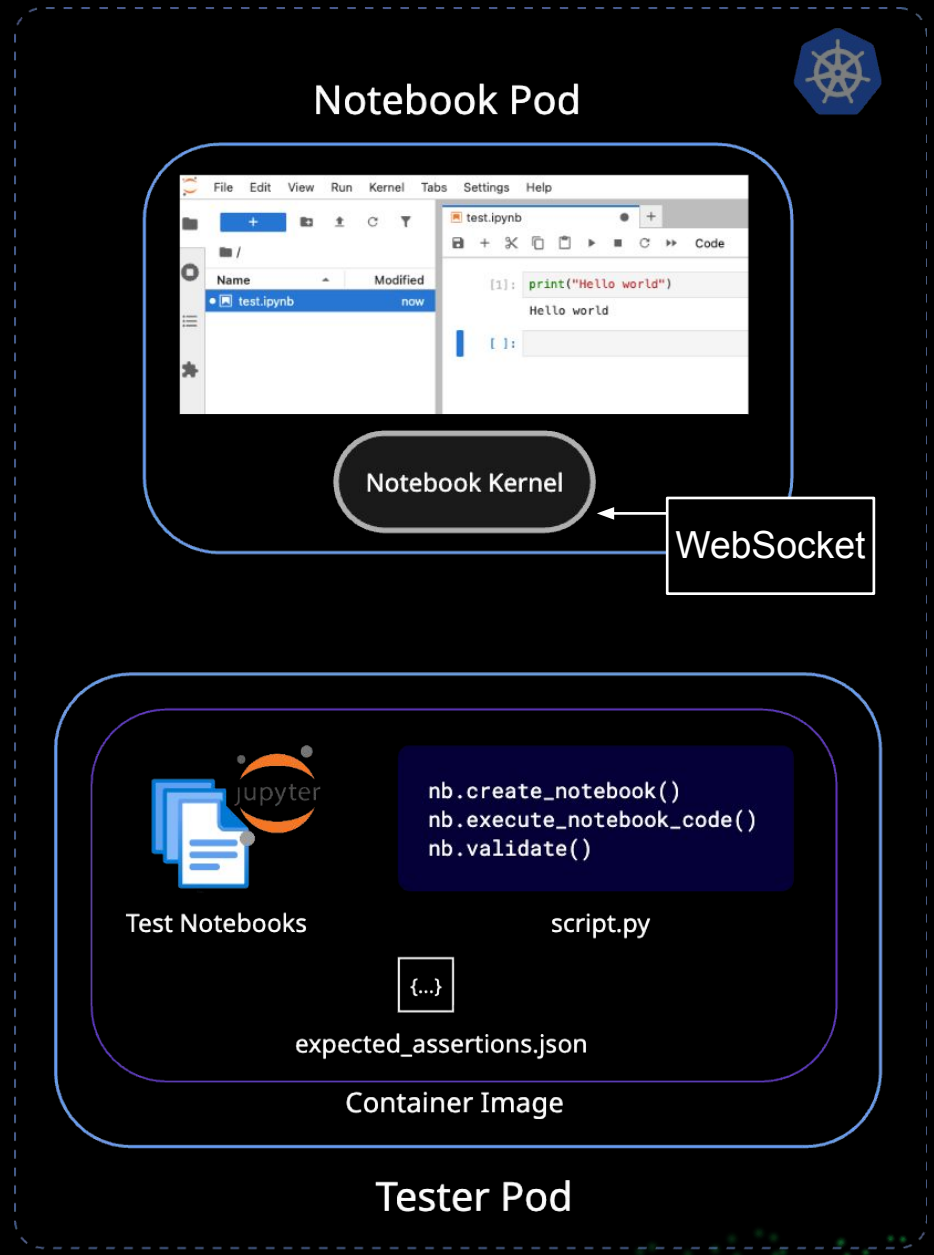
```
def create_notebook():
  requests.post(url/api/contents/{filename}, body)

def create_kernel():
  requests.post(url/api/kernels)

def kill_kernel(kernel_id):
  requests.delete(url/api/kernels/{kernel_id})

def execute_notebook_code():
  code = requests.get(url/api/contents/{notebook})
  kernel_id = create_kernel()
  ws = websocket("wss://url/api/kernels/{kernel_id}/channels")
  ws.send(code)

nb = NotebookTester(jupyter_server_url)
nb.create_notebook()
nb.execute_notebook_code()
nb.validate()
```



TechAtBloomberg.com

© 2025 Bloomberg Finance L.P. All rights reserved.

Tester Pod Pseudo Code

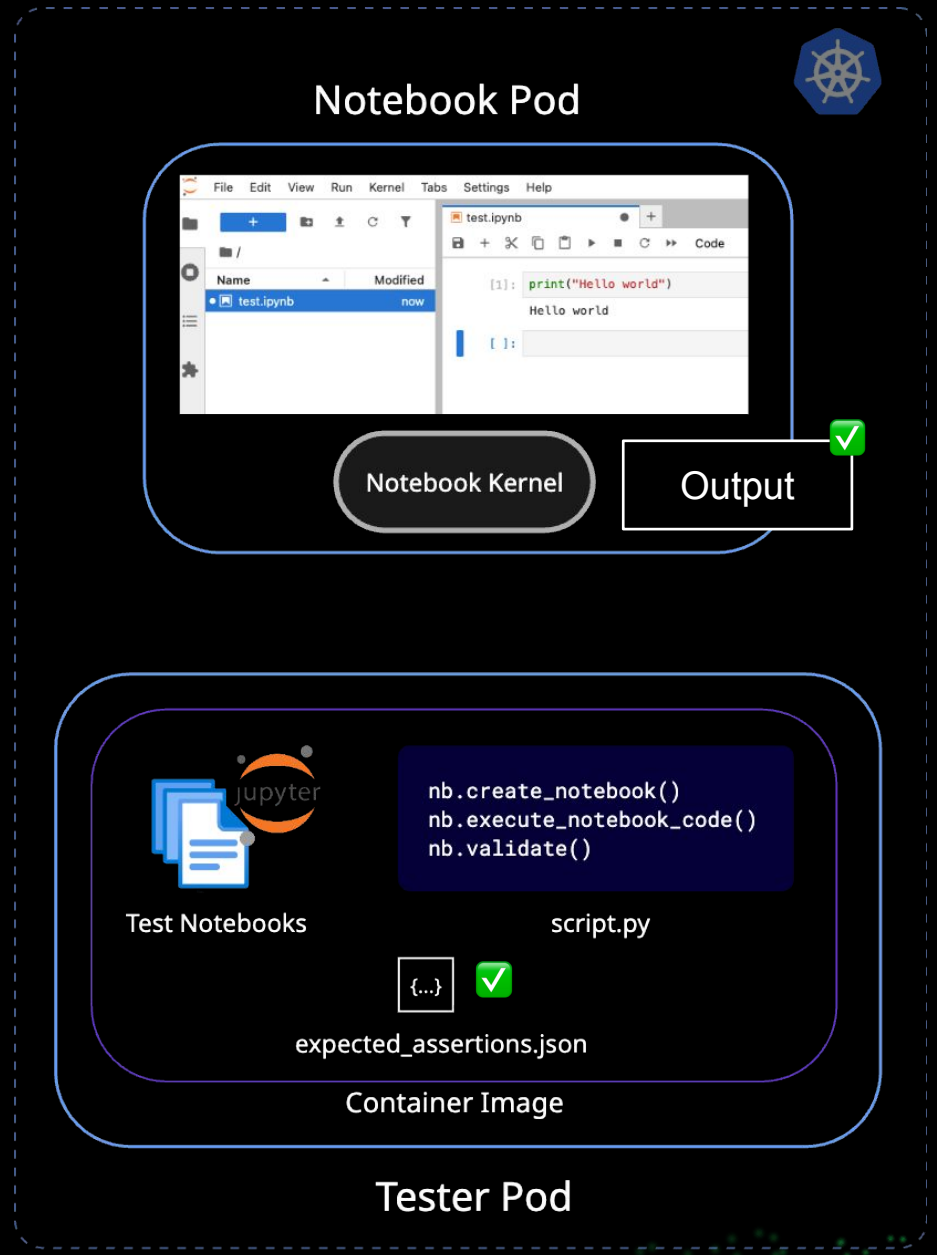
```
def create_notebook():
  requests.post(url/api/contents/{filename}, body)

def create_kernel():
  requests.post(url/api/kernels)

def kill_kernel(kernel_id):
  requests.delete(url/api/kernels/{kernel_id})

def execute_notebook_code():
  code = requests.get(url/api/contents/{notebook})
  kernel_id = create_kernel()
  ws = websocket("wss://url/api/kernels/{kernel_id}/channels")
  ws.send(code)

nb = NotebookTester(jupyter_server_url)
nb.create_notebook()
nb.execute_notebook_code()
nb.validate()
```



TechAtBloomberg.com

© 2025 Bloomberg Finance L.P. All rights reserved.

Tester Pod Pseudo Code

```
def create_notebook():
  requests.post(url/api/contents/{filename}, body)

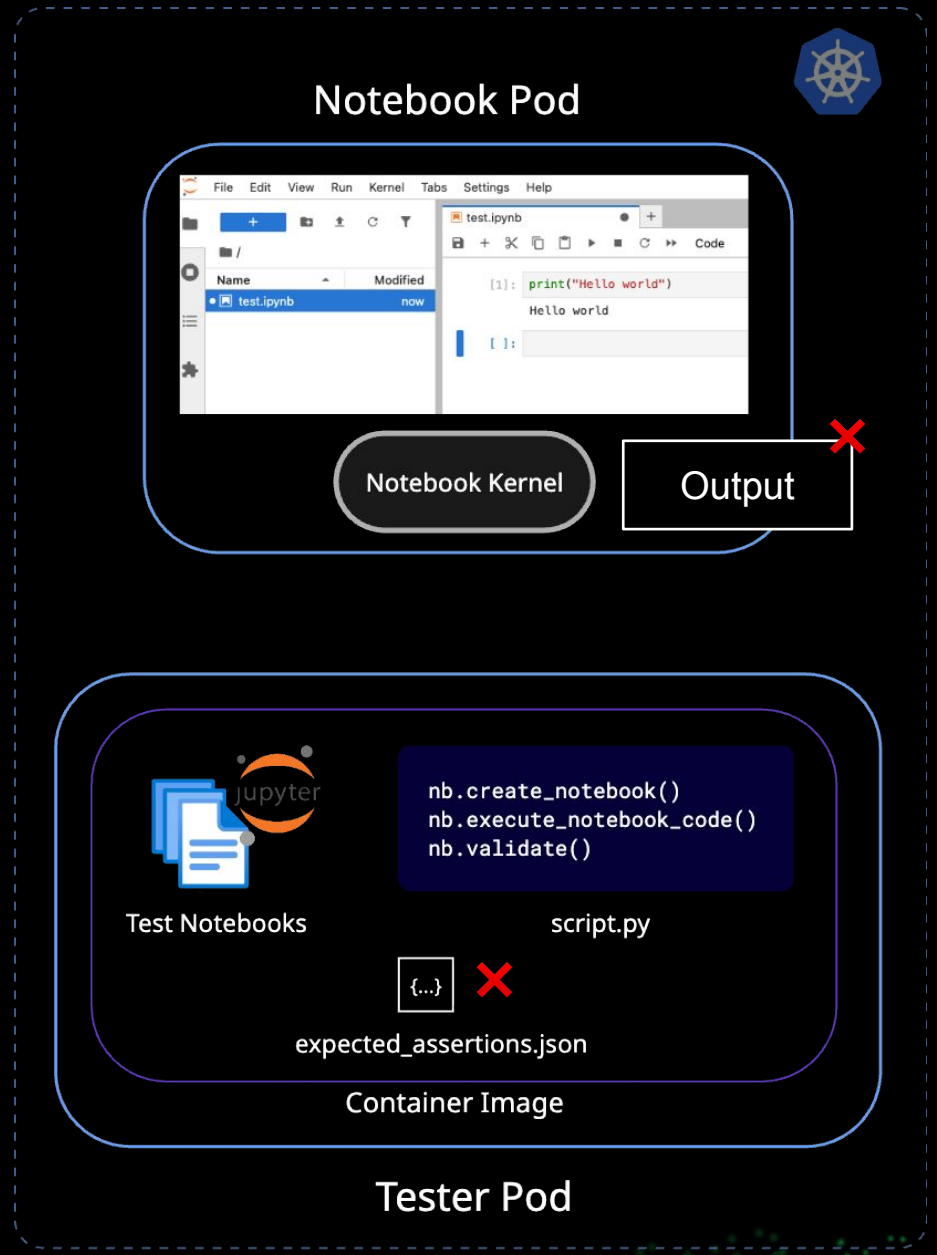
def create_kernel():
  requests.post(url/api/kernels)

def kill_kernel(kernel_id):
  requests.delete(url/api/kernels/{kernel_id})

def execute_notebook_code():
  code = requests.get(url/api/contents/{notebook})
  kernel_id = create_kernel()
  ws = websocket("wss://url/api/kernels/{kernel_id}/channels")
  ws.send(code)

nb = NotebookTester(jupyter_server_url)
nb.create_notebook()
nb.execute_notebook_code()
nb.validate()
```

If there is a failure, call the Slack API to ping the on-call contact for investigation



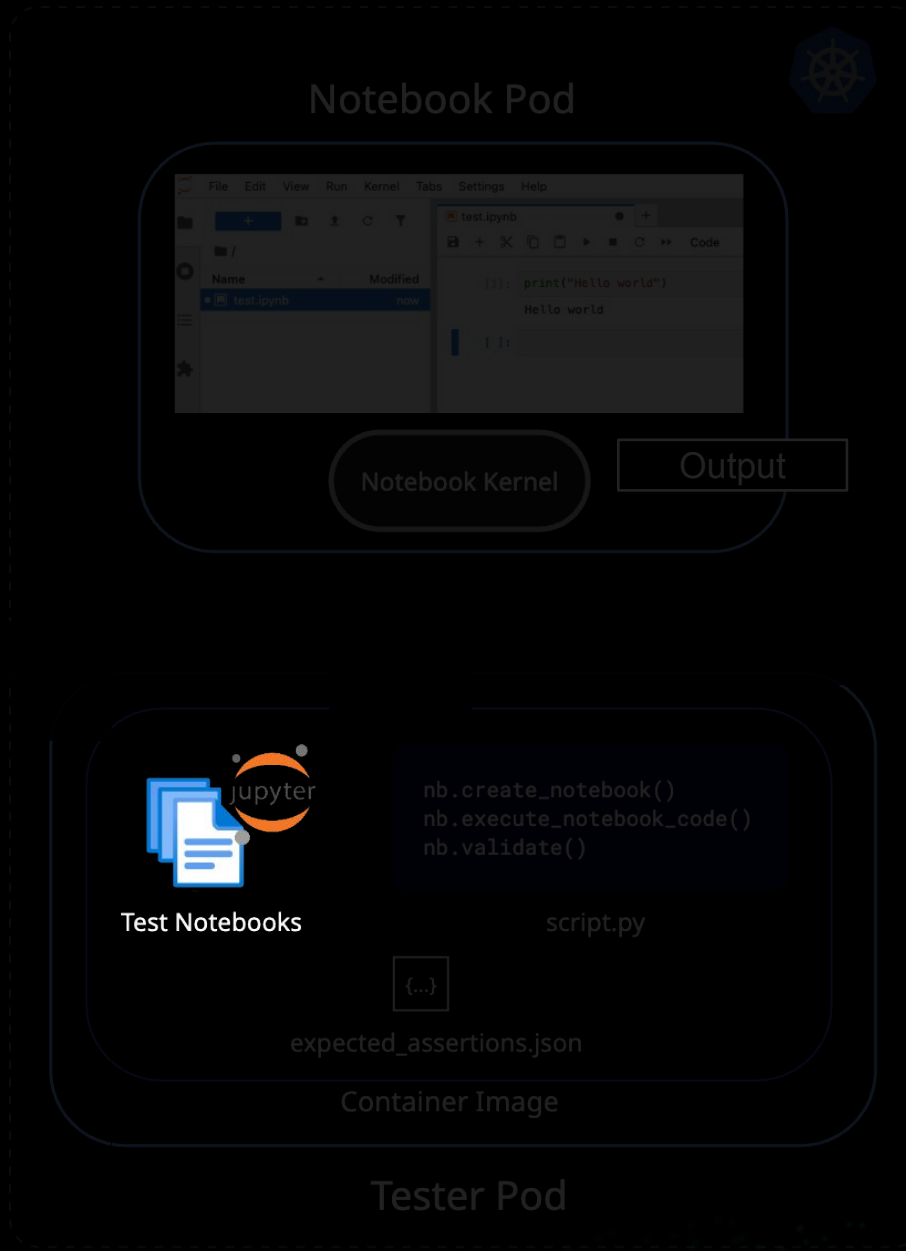
TechAtBloomberg.com

© 2025 Bloomberg Finance L.P. All rights reserved.

What are we checking?

Checklist

- Notebook job launch and accessibility
- Notebook and kernel creation
- Code execution
- GPU connectivity
- Models, dataset registry, and remote storage
- Connectivity to ML tools and internal services
- Jupyter features
- ML libraries: PyTorch, Hugging Face, Transformers
- LLM endpoints
- Credential management



TechAtBloomberg.com

© 2025 Bloomberg Finance L.P. All rights reserved.

Orchestrate with Argo, Execute with Jupyter API

🚀 Launch Notebook jobs with Argo Workflows

🤖 Spawn tester pods for Notebook creation, execution, and validation

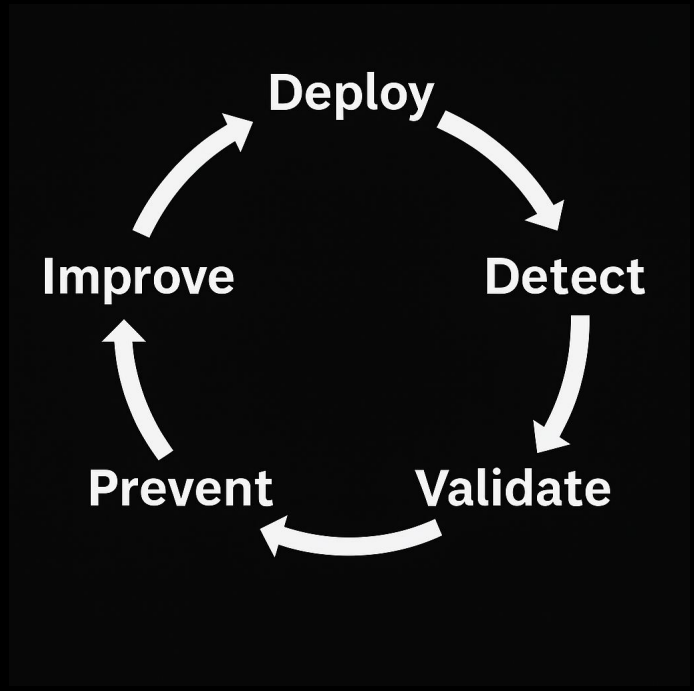
🔗 Use the Jupyter API for Notebook operations

⏰ Automate with cron schedules or deployment triggers

© 2025 Bloomberg Finance L.P. All rights reserved.

Why Automated Notebook Validation Matters

- Confidence in deployments
- Early detection of breakage
- Compatibility checks
- Regression prevention
- Save engineering efforts



© 2025 Bloomberg Finance L.P. All rights reserved.

How do we measure the platform’s health?

© 2025 Bloomberg Finance L.P. All rights reserved.

Bloomberg

Engineering

Defining Service Level Agreements/Objectives

Notebook Pod Initialization Time

Pod initialization (including image pull and setup) should complete within *N* seconds for at least 99% of launches

Notebook Launch Success Rate

At least 99% of notebook launches should successfully transition from the Initializing state to the Running state

© 2025 Bloomberg Finance L.P. All rights reserved.

SLOs and SLAs



Illustration only: These numbers aren't from production machines

$$\text{Success Rate} = \frac{\text{Good Events}}{\text{Bad Events}} \times 100$$

$$\text{Burn Rate} = \frac{\text{Errors Observed in Time Window}}{\text{Error Budget for that Time Window}}$$

$$\text{Error Budget} = 100\% - \text{SLO}$$

A high burn rate should trigger alerts for triaging

Watch Alexa Griffith & Ankita Chaudhari's KubeCon Europe 2025 talk (April 2, 2025): "***Dashboards & Dragons: Crafting SLOs To Tame the AI Platform Chaos***"

TechAtBloomberg.com

© 2025 Bloomberg Finance L.P. All rights reserved.

Examples

Some notebooks exceeded the pod initialization SLA

- **Root cause:** PyTorch installation from pip delayed start-up
- **Mitigation:** Pre-baked PyTorch into the Jupyter image for faster initialization

Notebook launch success rate decreased

- GPU errors increased in Argo-Jupyter API Validation Framework
- Correlation showed GPU notebooks were failing on launch
- **Mitigation:** Fixing GPU issues improved the success rate

© 2025 Bloomberg Finance L.P. All rights reserved.

Summary

🚀 Argo workflows for job scheduling

🚀 Jupyter API - Notebook creation, kernel control, and code execution

🚀 SLOs and SLAs

🚀 Continuous monitoring and observability

🚀 **Reliability is a core platform feature**

TechAtBloomberg.com

© 2025 Bloomberg Finance L.P. All rights reserved.

Thank you!

Questions?

We are hiring: https://www.bloomberg.com/careers

Bloomberg

Engineering

TechAtBloomberg.com

© 2025 Bloomberg Finance L.P. All rights reserved.