### **1. Confirm the GitHub App Is Installed and Has Access**

* Check that on Github the port app is installed on the GitHub organization.
* Verify that it has access to the correct repository.
* Review permissions it needs “Actions: Read and Write.”
* If the app is not installed or lacks repo access, Port cannot start any workflows.

### **2. Verify the Workflow File Exists and Is Discoverable**

* The workflow file must be inside .github/workflows/.
* The filename in Port must match exactly (including .yml or .yaml).
* The file needs to be in the repository’s default branch (main or master).
* Validate that the YAML is syntactically correct.

A missing or misnamed file will prevent GitHub from recognizing the workflow.

### **3. Check the Workflow Trigger Type**

* Port requires workflow\_dispatch to trigger workflows.

Correct trigger example:

on:

workflow\_dispatch:

inputs:

service\_name:

required: true

type: string

environment:

required: true

type: string

### **4. Validate Input Mapping Between Port and GitHub**

* In the Port action configuration, ensure that workflowInputs is defined.
* Each input expected by the workflow (service\_name, environment, etc.) must be mapped from the Port action’s inputs.

Example:

"workflowInputs": {

"service\_name": "{{ .inputs.service\_name }}",

"environment": "{{ .inputs.environment }}"

}

If this mapping is missing, the workflow will not start, and the action will appear stuck.

### **5. Confirm Organization, Repository, and Workflow Names**

* Double-check that the organization and repository names in the Port action match GitHub exactly.
* Even small typos (case sensitivity, extra characters) will prevent the action from finding the workflow.

### **6. Ensure Authentication for Port Updates**

* If the workflow runs but cannot update the status back in Port, check GitHub repository secrets.

Required secrets:  
  
 PORT\_CLIENT\_ID=...

PORT\_CLIENT\_SECRET=...

* These values come from Port’s **Settings → Credentials**.

### **7. Test Triggering the Workflow Directly via GitHub API**

To rule out whether the issue is in Port or GitHub:

curl -X POST \

-H "Authorization: token <GITHUB\_PAT\_TOKEN>" \

-H "Accept: application/vnd.github.v3+json" \

https://api.github.com/repos/ORG/REPO/actions/workflows/workflow.yml/dispatches \

-d '{"ref":"main","inputs":{"service\_name":"test","environment":"dev"}}'

* If this works, the workflow is healthy and the issue is in the Port configuration.
* If it doesn’t, then GitHub itself is rejecting the trigger (likely due to trigger type, inputs, or file location).

### **8. Inspect GitHub Actions Tab and Workflow Logs**

* Go to the repository’s **Actions tab** and check whether runs are appearing at all.
* If they do appear but fail early, review the logs — this often reveals missing inputs, authentication errors, or YAML issues.

### **9. Use Debug Logging in the Workflow**

For clarity, add a debug step to print all received inputs:

- name: Debug inputs

run: |

echo "Inputs: ${{ toJson(github.event.inputs) }}"

echo "Event: ${{ github.event\_name }}"

This confirms whether GitHub is receiving what Port sent.

## **Summary**

The systematic order is:

1. App installation and permissions
2. Workflow file location and branch
3. Correct trigger (workflow\_dispatch)
4. Input mapping (workflowInputs)
5. Exact org/repo/workflow naming
6. Port authentication secrets
7. Manual API test to isolate GitHub vs. Port
8. GitHub Actions tab and logs
9. Debug logging inside workflow