# **Portfolio Submission**

## **Exercise 2 – Steps Taken: Set Up Port ↔ Jira ↔ GitHub**

**1) Installed and Connected the GitHub App**

* Installed the Port GitHub App on the correct organization.
* Granted access to selected repositories with “Actions: Read and Write” permissions.
* Confirmed Link to the installation on Port under **Datasources** and verified repository entities appeared.

**2) Created Jira Project with Correct Template**

* Created a Jira project using the **Software development → Scrum → Company-managed** template.
* Confirmed company-managed features (workflows, schemes, and fields) were available.

**3) Created Jira Components to Match GitHub Repositories**

* Defined at least two Jira components, each named to match a GitHub repository.
* Verified that issues could be tagged with the correct component.

**4) Deployed the Jira Integration in Port**

* In Port: **Data Sources → Add Data Source → Jira**.
* Chose **Scheduled** sync mode with **GitHub Workflow** deployment.
* Generated Jira API tokens and added them to GitHub repository secrets (atlassianUserEmail, atlassianUserToken).
* Created .github/workflows/jira-integration.yaml with the following workflow:

name: Jira Exporter Workflow

on:

workflow\_dispatch:

schedule:

- cron: '0 \*/1 \* \* \*' # Runs every hour

jobs:

run-integration:

runs-on: ubuntu-latest

timeout-minutes: 30

steps:

- name: Run Jira Integration

uses: port-labs/ocean-sail@v1

with:

type: jira

port\_client\_id: ${{ secrets.PORT\_CLIENT\_ID }}

port\_client\_secret: ${{ secrets.PORT\_CLIENT\_SECRET }}

port\_base\_url: "https://api.port.io"

config: |

jira\_host: "https://jasonnwakaeze.atlassian.net"

atlassian\_user\_email: ${{ secrets.atlassianUserEmail }}

atlassian\_user\_token: ${{ secrets.atlassianUserToken }}

initialize\_port\_resources: false # Remove this line on first run

* Verified both manual (workflow\_dispatch) and scheduled runs populated Jira issues/components in Port.

**5) Verified Jira Issue → Repository Relation**

* In Port blueprints, established a relation from **Jira Issue → Repository** based on component-to-repo mapping.
* Confirmed issues across multiple components linked correctly to their repositories.

## **Exercise 3 – Scorecard**

**1) Created Aggregation Property**

* In the Repository blueprint, added a new **aggregation property**.
* Configured it to count all related **Pull Requests** where the status is open.

**2) Applied Scorecard Logic**

* Built a Scorecard to classify repositories by open PR count:  
  + **Gold**: fewer than 5 open PRs
  + **Silver**: fewer than 10 open PRs
  + **Bronze**: fewer than 15 open PRs

**3) Tested Against Real Repositories**

* Compared calculated values against actual open PR counts in GitHub.
* Confirmed repositories were correctly categorized as Gold, Silver, or Bronze.

**Outcome**

* The Scorecard dynamically reflects repository health and surfaces PR backlog risk.

## 

## 

## 

## **Exercise 4 – Troubleshooting a Port Self-Service Action Stuck in “IN PROGRESS”**

**Scenario**: A self-service action was configured to trigger a GitHub workflow but remained stuck in “IN PROGRESS.”

**Steps Taken to Diagnose and Resolve**

1. **Verified GitHub App Installation**
   * Confirmed the Port GitHub App was installed on the organization, with repository access and “Actions: Read and Write” permission.
2. **Checked Workflow File**
   * Ensured the workflow existed at .github/workflows/ on the default branch.
   * Validated the YAML syntax and filename.
3. **Corrected Workflow Trigger**
   * Identified that repository\_dispatch does not work with Port.
   * Updated the workflow to use workflow\_dispatch with defined inputs.
4. **Validated Input Mapping**
   * Confirmed the Port action configuration included workflowInputs that mapped to the workflow’s inputs (service\_name, environment).
5. **Checked Org/Repo/Workflow Naming**
   * Verified the names matched exactly between Port and GitHub.
6. **Confirmed Authentication**
   * Ensured PORT\_CLIENT\_ID and PORT\_CLIENT\_SECRET were added as GitHub repository secrets so the workflow could update run status in Port.
7. **Manual API Test**
   * Triggered the workflow directly using the GitHub REST API to confirm it could start independently of Port.
8. **Reviewed Logs in GitHub Actions Tab**
   * Verified whether the workflow started and, if so, whether it failed early due to missing inputs or secrets.
9. **Added Debugging**
   * Inserted a debug step to print github.event.inputs for clarity.

**Outcome**

* The root cause was a mismatch in workflow trigger (repository\_dispatch instead of workflow\_dispatch) and missing workflowInputs mapping.
* After corrections, the self-service action triggered successfully and reported back completion status.