Support Engineer Challenge

Overview

As a support engineer for Port you need to help customers understand how to use Port and its various features, as well as help them debug issues that they encounter when using Port, its API and the various offered integrations

Assumptions & Baseline

- As part of this challenge, you will receive several real-life examples of questions, queries and issues encountered by Port's customers. You will also receive tasks asking you to setup several Port integrations and connect them to your Port account, or update Port using automations based on data that exists in the catalog
- The goal is to see how you answer customer's questions and also that you are able to replicate the steps taken by a user of Port when he encounters an issue. You will be evaluated according to a list including (but not limited to) the following parameters
    - For questions that require a response to a customer
        - Answer correctness
        - Answer phrasing and articulation
        - Answer precision (answering what the customer needed help with, without adding too much unnecessary information)
    - For questions that require a solution implementation
        - Solution result
        - Solution design and explanation

Resources

For this assignment, you can make use of the following resources:

- Port's developer portal - https://app.getport.io
- Port's documentation - https://docs.getport.io
- Port's REST API reference - https://api.getport.io/static/index.html
- Port's GitHub app
- Port's Ocean documentation - https://ocean.getport.io
    - Ocean's Jira integration
- Port's open community slack - https://www.getport.io/community
- Feel free to use Google and other similar tools

Notes

- For this exercise you will need a Port account
- To access the example questions taken from the Slack community, you will need to join the community: https://www.getport.io/community
- For this exercise you will need a way to deploy Port Ocean integrations, you can use minikube, k3s, or any other target that supports deploying to Kubernetes by using Helm
- Port's UI will provide you with instructions on how to deploy the Ocean integration for Jira, you can access the installation instructions from the data sources tab inside the Settings page
- Please solve all exercises in the same Port account, you can use the same blueprints (for example, the service blueprint which is used in several exercises) for multiple exercises, or create new blueprints for each exercise as needed
- During the review of the exercise, you will be asked to explain and demonstrate at least one of the exercises in English. In addition, you are expected to successfully explain core Port concepts such as blueprint, entity and self-service action

Exercise #1

The following exercises describe common scenarios customers encounter when trying to take an API response and transform it using JQ. For this exercise, only a proper JQ pattern with an explanation is required.
Given the following K8s deployment object snippet, **provide** JQ patterns that will extract the following information:

1. The current replica count
2. The deployment strategy
3. The "service" label of the deployment concatenated with the "environment" label of the deployment, with a hyphen (-) in the middle

Given the following Jira API issue response, provide a JQ pattern that will extract all of the issue IDs (for example SAMPLE-123) for all subtasks, in an **array**

Tip: It is recommended to use JQPlay to check your answers

Exercise #2

A customer is wondering what SSO providers we offer an integration with, he mentions that they are using Okta for SSO and asking for our help configuring his SSO with Okta.

Provide a comprehensive answer, in addition, provide him with the information he requires from Port's team in order to properly configure Okta and ask him in return for the information we require from his side (for the information we provide, just provide a sample, doesn't have to be real data since there is no real customer in this instance)

Limit: Intro + 2 paragraphs

Exercise #3

- Create a Port organization if you don't have one already.
- Install Port's GitHub app
- Install Port's Ocean integration for Jira (you can use the free tier version of Jira)
- Update Port's data model (from the builder page in Port) with a relation from "Jira Issue" to "Service" (The service blueprint will be created automatically when you install the GitHub app during the onboarding)
- Create components in Jira matching the GitHub repositories in your GitHub account (2 repositories, for example for Frontend and Backend are enough)
- Update the mapping used by the Jira integration in Port, such that when a component is added to a Jira issue, the component is used to relate the Jira Issue to the relevant GitHub repository in the service blueprint in Port
    - Assume for this assignment that a Jira issue can be related to multiple components (and as such, to multiple GitHub repositories)

Exercise #4

A customer is trying to configure a self-service action that triggers a GitHub workflow, he says that the workflow he expected is not being triggered, and that the self-service action in Port just stays in "IN PROGRESS" status indefinitely.

Offer some troubleshooting steps he can go through to verify that he did everything required to trigger a GitHub workflow from Port properly (think about all of the different requirements and potential mistakes the customer might make when configuring his self-service action)

This exercise is meant to display debugging intuition and troubleshooting. We suggest trying to install the GitHub App and triggering a self-service action to understand how the real process looks and some points where a user might make mistakes or fail.

Limit: Intro + 3 troubleshooting steps (1 sentence/clarification question to the customer each)

Exercise #5

This exercise is based on the following question by a Port customer.

Introduction

A customer has **services** mapped to his Port account. He wants to add a **scorecard** to his service blueprint that tracks the number of open PRs for a repository with the following logic:

- < 5 PRs - Gold
- < 10 PRs - Silver
- < 15 PRs - Bronze

Assignment

- Create a property that will count the number open PRs a repository has
- Create a scorecard to the blueprint that will include the logic outlined above

Tip: for this exercise, once the correct data is inside Port, the solution can be implemented using Port alone.

Exercise #6

This exercise is based on the following question by a Port customer.

Introduction

A customer has a service blueprint and a framework blueprint, with a relation **service** -> used **frameworks** (a many relation), his framework blueprint has a **State** property, the available values for the state property are **Active** / **EOL**.

He wants to add to his service blueprint a property called **Number of EOL packages**.

Assignment

- Create the matching Port environment (blueprints, relations, entities) - you can use mock data (of course except for the number of EOL packages property which needs to be properly updated by the script)
    - For the framework entities - you can manually create a few mock entities that have the **State** property set to either **Active / EOL**.
- Write a script in Python that queries the different service entities and frameworks, and updates the value of the number of EOL packages property on each service entity with the correct value.