**Support Engineer Challenge**

**Victor Hugo Vergara Cervantes**

**12/29/2025**

## Exercise #1

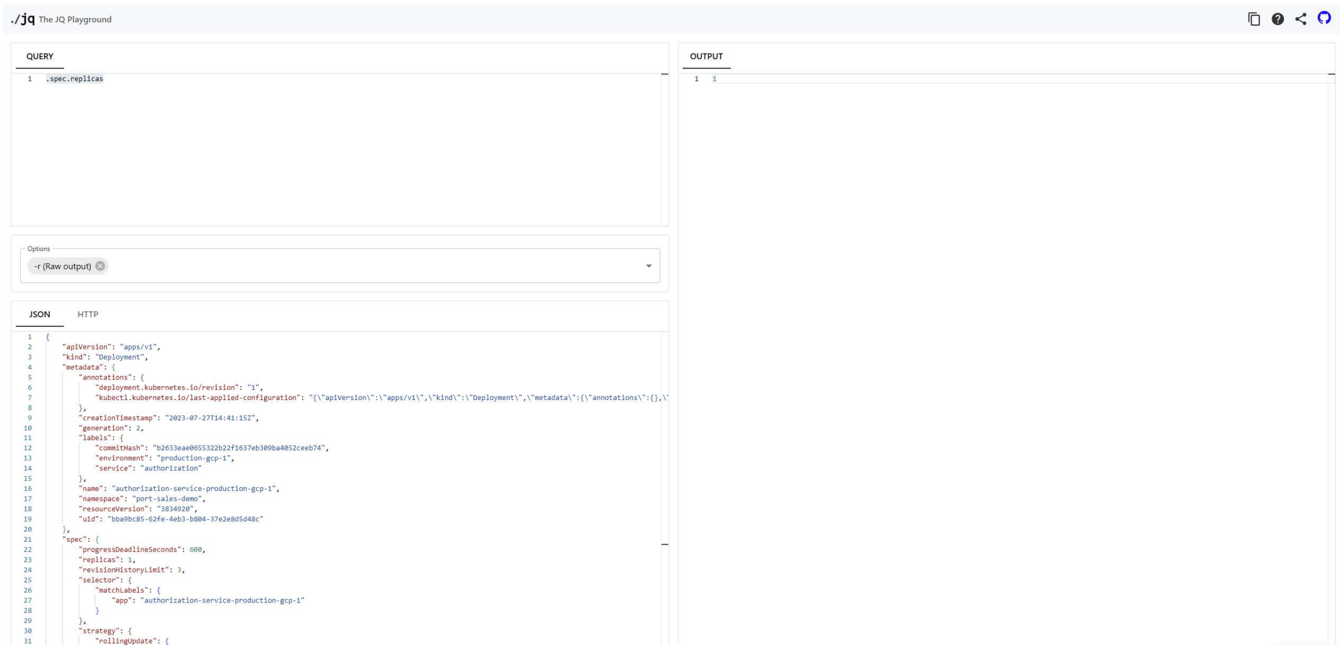
The following exercises describe common scenarios customers encounter when trying to take an API response and transform it using JQ. For this exercise, only a proper JQ pattern with an explanation is required.

Given the following K8s deployment object snippet, **provide** JQ patterns that will extract the following information:

1. The current replica count

ANSWER

jq ".spec.replicas"



```
.spec.replicas
```

Options: -r (Raw output)

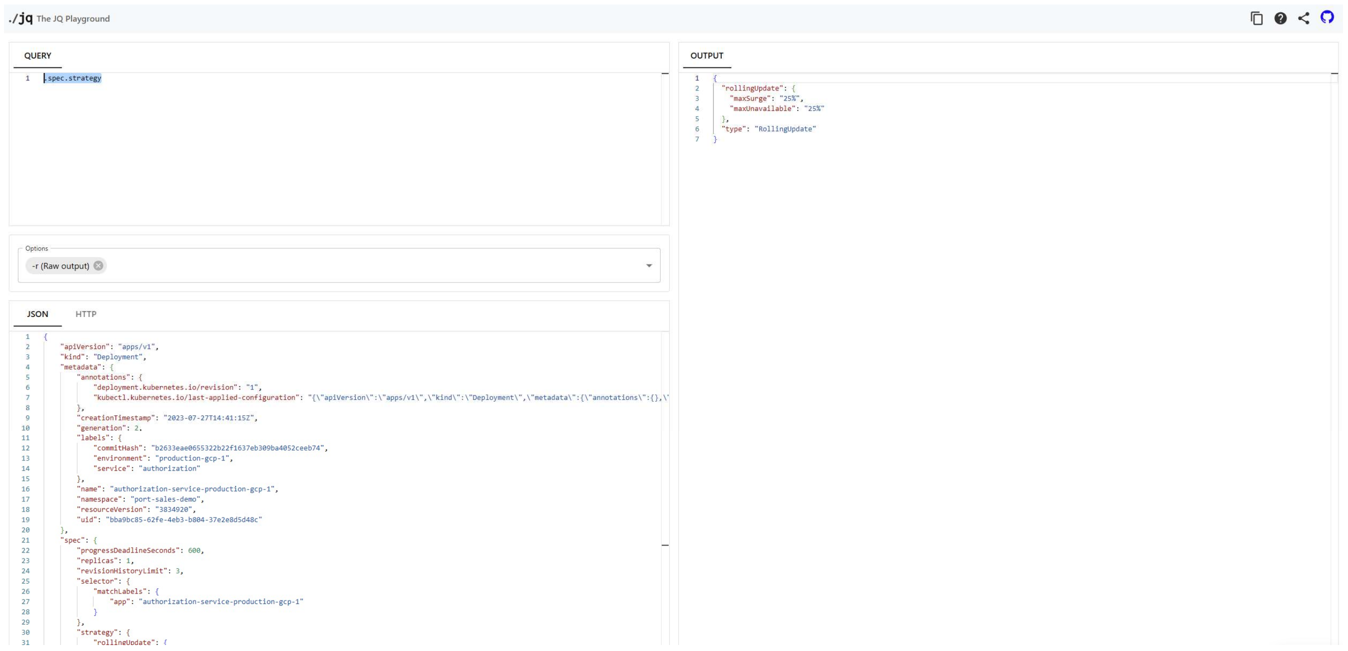
```
1
```

```
{
    "apiVersion": "apps/v1",
    "kind": "Deployment",
    "metadata": {
        "annotations": {
            "deployment.kubernetes.io/revision": "1",
            "kubectl.kubernetes.io/last-applied-configuration": "{\"apiVersion\":\"apps/v1\",\"kind\":\"Deployment\",\"metadata\":{\"annotations\":{},\'
        },
        "creationTimestamp": "2023-07-27T14:41:15Z",
        "generation": 2,
        "labels": {
            "commitHash": "b2633eae0655322b22f1637eb309ba4052ceeb74",
            "environment": "production-gcp-1",
            "service": "authorization"
        },
        "name": "authorization-service-production-gcp-1",
        "namespace": "port-sales-demo",
        "resourceVersion": "3834920",
        "uid": "bba9bc85-62fe-4eb3-b804-37e2e8d5d48c"
    },
    "spec": {
        "progressDeadlineSeconds": 600,
        "replicas": 1,
        "revisionHistoryLimit": 3,
        "selector": {
            "matchLabels": {
                "app": "authorization-service-production-gcp-1"
            }
        },
        "strategy": {
            "rollingUpdate": {
```

2. The deployment strategy

ANSWER

jq ".spec.strategy"



./jq The JQ Playground

QUERY

```
.spec.strategy
```

OUTPUT

```
{
  "rollingUpdate": {
    "maxSurge": "25%",
    "maxUnavailable": "25%"
  },
  "type": "RollingUpdate"
}
```

Options: -r (Raw output)

JSON | HTTP

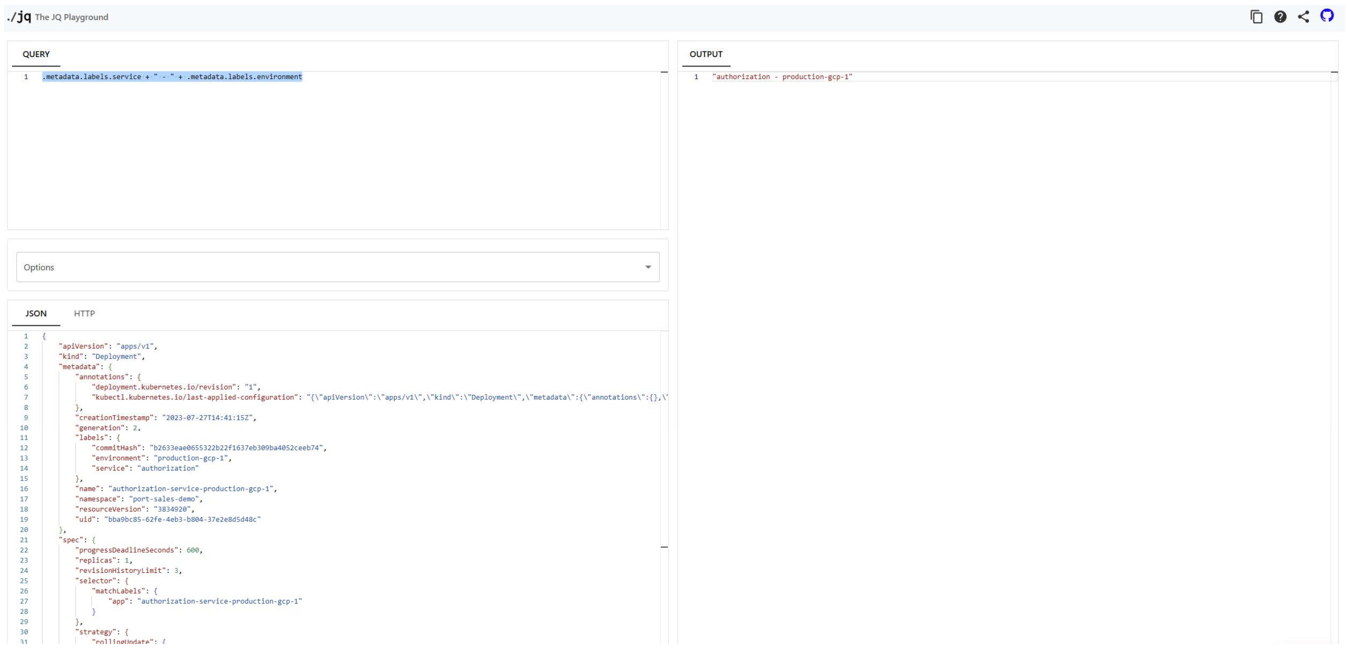
```
{
    "apiVersion": "apps/v1",
    "kind": "Deployment",
    "metadata": {
        "annotations": {
            "deployment.kubernetes.io/revision": "1",
            "kubectl.kubernetes.io/last-applied-configuration": "{\"apiVersion\":\"apps/v1\",\"kind\":\"Deployment\",\"metadata\":{\"annotations\":{},\'
        },
        "creationTimestamp": "2023-07-27T14:41:15Z",
        "generation": 2,
        "labels": {
            "commitHash": "b2633eae0655322b22f1637eb309ba4052ceeb74",
            "environment": "production-gcp-1",
            "service": "authorization"
        },
        "name": "authorization-service-production-gcp-1",
        "namespace": "port-sales-demo",
        "resourceVersion": "3834920",
        "uid": "bba9bc85-62fe-4eb3-b804-37e2e8d5d48c"
    },
    "spec": {
        "progressDeadlineSeconds": 600,
        "replicas": 1,
        "revisionHistoryLimit": 3,
        "selector": {
            "matchLabels": {
                "app": "authorization-service-production-gcp-1"
            }
        },
        "strategy": {
            "rollingUpdate": {
```

3. The “service” label of the deployment concatenated with the “environment” label of the

deployment, with a hyphen (-) in the middle

ANSWER

jq “.metadata.labels.service + "- " + .metadata.labels.environment”



./jq The JQ Playground

QUERY

```
.metadata.labels.service + " - " + .metadata.labels.environment
```

OUTPUT

```
"authorization - production-gcp-1"
```

Options

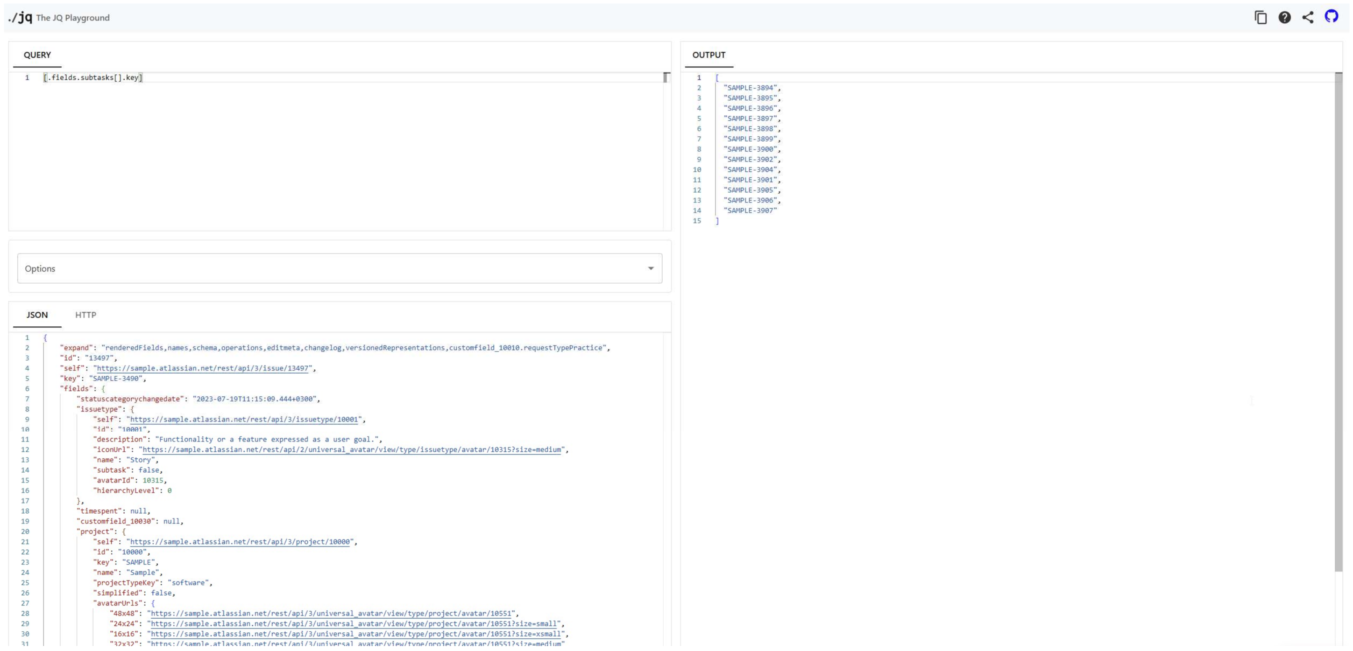
JSON | HTTP

```
{
    "apiVersion": "apps/v1",
    "kind": "Deployment",
    "metadata": {
        "annotations": {
            "deployment.kubernetes.io/revision": "1",
            "kubectl.kubernetes.io/last-applied-configuration": "{\"apiVersion\":\"apps/v1\",\"kind\":\"Deployment\",\"metadata\":{\"annotations\":{},\'
        },
        "creationTimestamp": "2023-07-27T14:41:15Z",
        "generation": 2,
        "labels": {
            "commitHash": "b2633eae0655322b22f1637eb309ba4052ceeb74",
            "environment": "production-gcp-1",
            "service": "authorization"
        },
        "name": "authorization-service-production-gcp-1",
        "namespace": "port-sales-demo",
        "resourceVersion": "3834920",
        "uid": "bba9bc85-62fe-4eb3-b804-37e2e8d5d48c"
    },
    "spec": {
        "progressDeadlineSeconds": 600,
        "replicas": 1,
        "revisionHistoryLimit": 3,
        "selector": {
            "matchLabels": {
                "app": "authorization-service-production-gcp-1"
            }
        },
        "strategy": {
            "rollingUpdate": {
```

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**

ANSWER

jq “[.fields.subtasks[].key]”



./jq The JQ Playground

QUERY

```
[.fields.subtasks[].key]
```

OUTPUT

```
[
  "SAMPLE-3894",
  "SAMPLE-3895",
  "SAMPLE-3896",
  "SAMPLE-3897",
  "SAMPLE-3898",
  "SAMPLE-3899",
  "SAMPLE-3900",
  "SAMPLE-3902",
  "SAMPLE-3904",
  "SAMPLE-3901",
  "SAMPLE-3905",
  "SAMPLE-3906",
  "SAMPLE-3907"
]
```

Options

JSON HTTP

```
{
    "expand": "renderedFields,names,schema,operations,editmeta,changelog,versionedRepresentations,customfield_10010.requestTypePractice",
    "id": "13497",
    "self": "https://sample.atlassian.net/rest/api/3/issue/13497",
    "key": "SAMPLE-3490",
    "fields": {
        "statuscategorychangedate": "2023-07-19T11:15:09.444+0300",
        "issuetype": {
            "self": "https://sample.atlassian.net/rest/api/3/issuetype/10001",
            "id": "10001",
            "description": "Functionality or a feature expressed as a user goal.",
            "iconUrl": "https://sample.atlassian.net/rest/api/2/universal_avatar/view/type/issuetype/avatar/10315?size=medium",
            "name": "Story",
            "subtask": false,
            "avatarId": 10315,
            "hierarchyLevel": 0
        },
        "timespent": null,
        "customfield_10030": null,
        "project": {
            "self": "https://sample.atlassian.net/rest/api/3/project/10000",
            "id": "10000",
            "key": "SAMPLE",
            "name": "Sample",
            "projectTypeKey": "software",
            "simplified": false,
            "avatarUrls": {
                "48x48": "https://sample.atlassian.net/rest/api/3/universal_avatar/view/type/project/avatar/10551",
                "24x24": "https://sample.atlassian.net/rest/api/3/universal_avatar/view/type/project/avatar/10551?size=small",
                "16x16": "https://sample.atlassian.net/rest/api/3/universal_avatar/view/type/project/avatar/10551?size=xsmall",
                "32x32": "https://sample.atlassian.net/rest/api/3/universal_avatar/view/type/project/avatar/10551?size=medium"
```

**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

ANSWER

Hi:

Port does offer several SSO Integrations that will allow you to assign permissions and roles using your users and teams. Here is a list of the supported SSO Integrations.

- AzureAD
- Okta
- JumpCloud
- Google Workspace
- OneLogin

You can either configure your Integration between Port and Okta using SAML or OIDC Protocol:

If you choose SAML Protocol, you can find the steps in this link:

- https://docs.port.io/sso-rbac/sso-providers/saml/okta/

Please fill out the following SAML settings in the Configure SAML Page:

- Single sign on URL: https://auth.getport.io/login/callback?connection={CONNECTION_NAME}
- Audience URI (SP Entity ID): urn:auth0:port-prod:{CONNECTION_NAME}

Also to secure your SAML integration you must generate a certificate in PEM format and provide it to Port along with the Identity Provider metadata URL which is available in the Sign On Tab.

If you choose OIDC Protocol you can find the steps in this other link:

- https://docs.port.io/sso-rbac/sso-providers/oidc/okta/

Please make sure you configure the proper redirect URIs that matches your account region

- EU organizations: https://auth.getport.io/login/callback
- US organizations: https://auth.us.getport.io/login/callback

Also provide Port with your ClientID and Okta Domain that will be in the format "{YOUR_COMPANY_NAME}.okta.com"

The authorization Endpoint should be set based on the account region, as an example:

- EU Region: https://auth.getport.io/authorize?response_type=token&client_id=96IeqL36Q0UIBxIfV1oqOkDWU6UslfDj&connection={CONNECTION_NAME}&redirect_uri=https%3A%2F%2Fapp.getport.io
- US Region: https://auth.us.getport.io/authorize?response_type=token&client_id=4lHUry3Gkds317lQ3J

cgABh0JPbT3rWx&connection={CONNECTION_NAME}&redirect_uri=https%3A%2F%2Fapp.us.getport.io

Please, if you have any further questions don't hesitate to ask me. Or if needed we can make a call to walk this through.

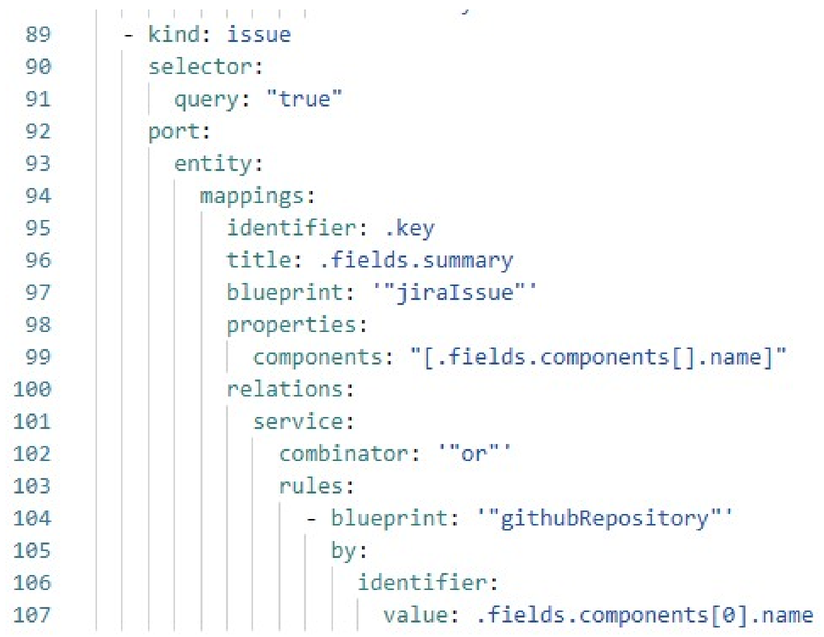
Thanks, greetings

**Exercise #3**

- Create a Port organization if you don't have one already.
- Install Port's GitHub app
- Create a Jira account
    - After creating the account, please create a new project, use the "software development" category and the "scrum" template, make the project a "company managed project" to get access to the components feature of Jira (you will see the components option in the sidebar on the left side of your Jira board).
    - When using Jira's components feature, be sure to use Jira components (and not Atlassian Compass components)
- Install Port's Ocean integration for Jira (you can use the free tier version of Jira)
    - You can choose to use either the "Real Time & Always on" or the "Scheduled" version of the Jira integration.
- 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 (for example, if you have a repository called "my-project", you will have a Jira component called "my-project". Please create at least 2 Jira components for 2 matching GitHub repositories)
- 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)

ANSWER

With this mapping configuration im defining a relation that uses the component name to find the GitHub repository entity associated.



```yaml
- kind: issue
  selector:
    query: "true"
  port:
    entity:
      mappings:
        identifier: .key
        title: .fields.summary
        blueprint: '"jiraIssue"'
        properties:
          components: "[.fields.components[].name]"
        relations:
          service:
            combinator: '"or"'
            rules:
              - blueprint: '"githubRepository"'
                by:
                  identifier:
                    value: .fields.components[0].name
```

**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)

ANSWER

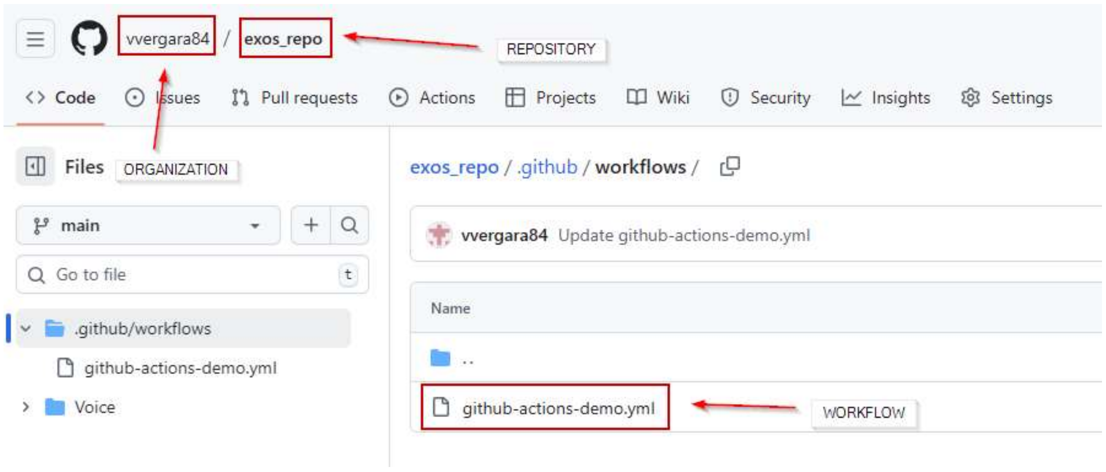
Hi:

Following up with your issue reported, can you please help me confirm that you have set up properly the GitHub Backend configuration.

Please make sure that:

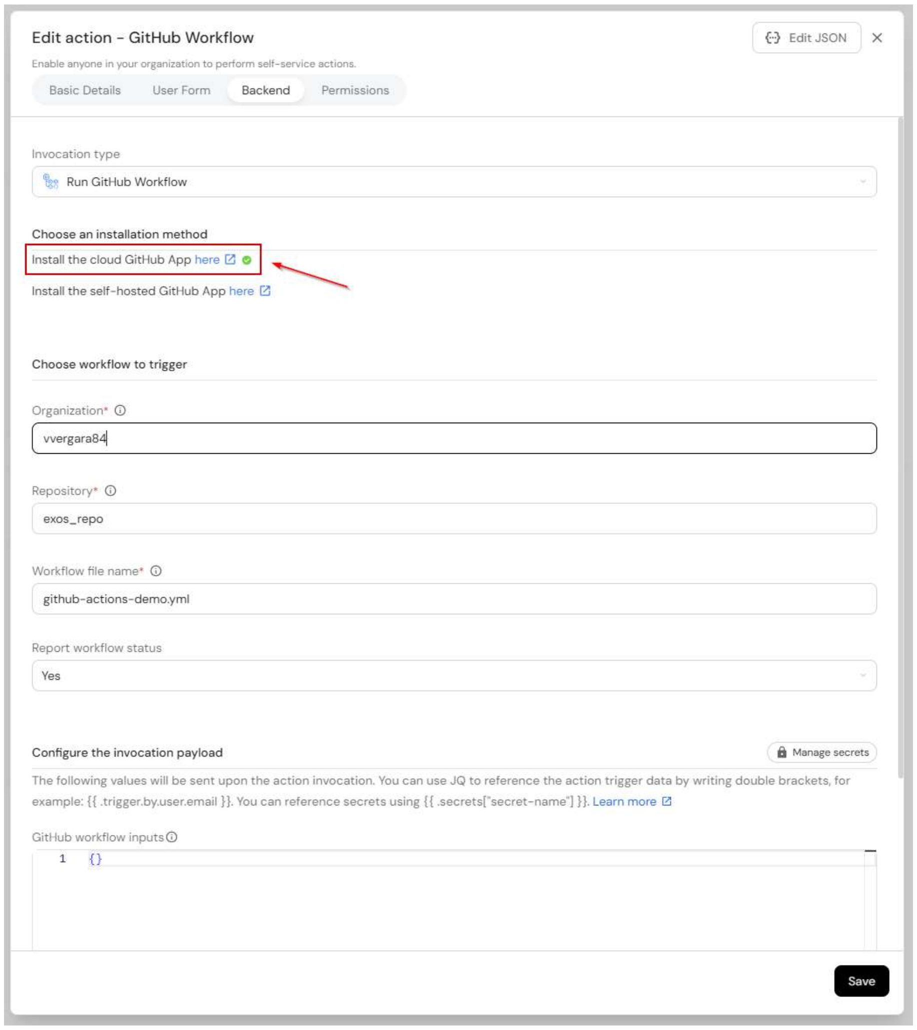
- You are using the correct Backend Type
    - Run GitHub Workflow
- Make sure that the configuration matches GitHub’s Workflow configuration

- Organization - The name of your GitHub organization/user, can be found in the organization/user URL. For example- "port-labs" for https://github.com/port-labs
- Repository- The name of the GitHub repository that contains the target workflow
- Workflow File Name- The name of the file (under /.github/workflows/) that defines the target workflow, for example- myworkflow.yml



¿Does your workflow in Github has enabled the workflow_dispatch event type?

¿Have you installed the cloud GitHub App? You should see a green check mark in the Port Action Configuration.



Edit action - GitHub Workflow

Enable anyone in your organization to perform self-service actions.

Basic Details | User Form | Backend | Permissions

Invocation type: Run GitHub Workflow

Choose an installation method

Install the cloud GitHub App here

Install the self-hosted GitHub App here

Choose workflow to trigger

Organization*: vvergara84

Repository*: exos_repo

Workflow file name*: github-actions-demo.yml

Report workflow status: Yes

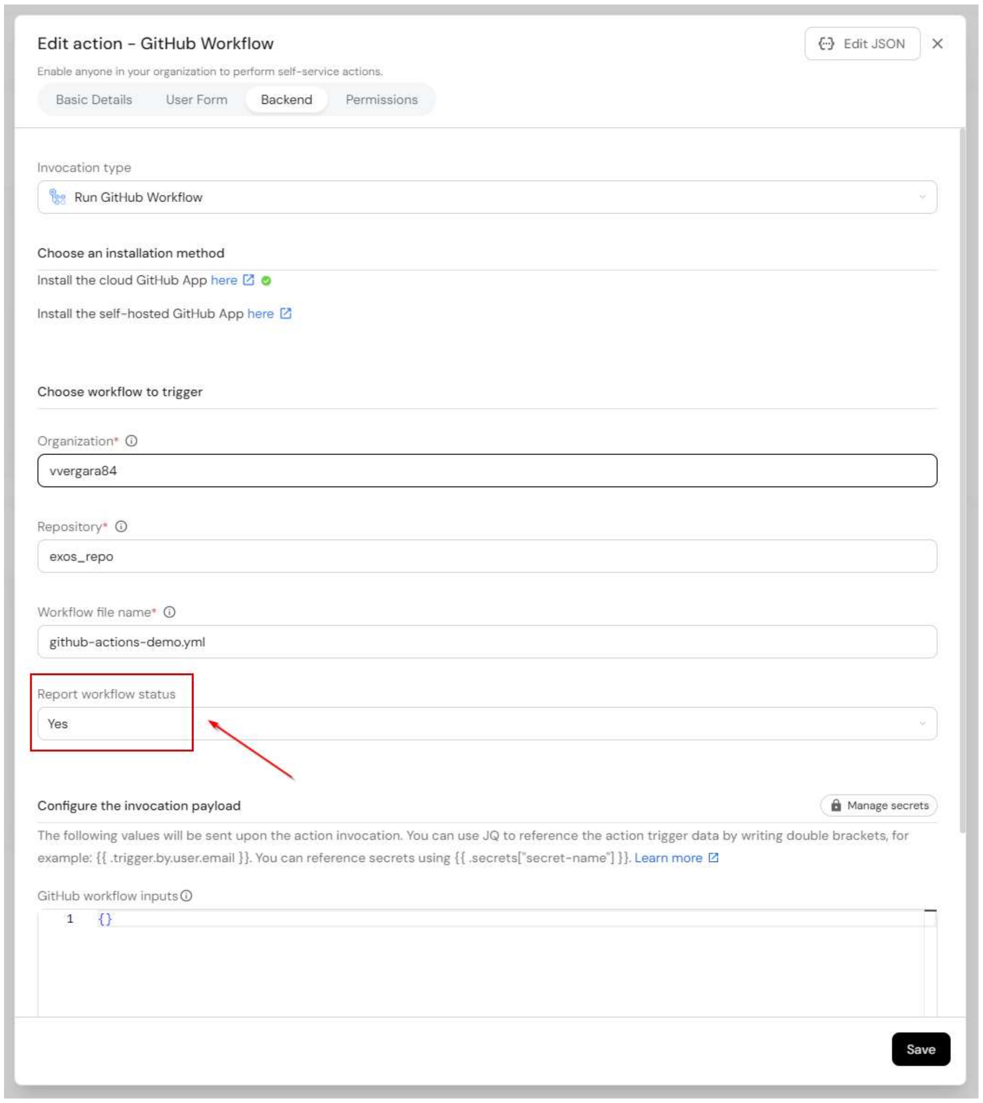
Configure the invocation payload

The following values will be sent upon the action invocation. You can use JQ to reference the action trigger data by writing double brackets, for example: {{ .trigger.by.user.email }}. You can reference secrets using {{ .secrets["secret-name"] }}. Learn more

GitHub workflow inputs

```
{}
```

When you execute the action are you getting any exception in the Summary. If nothing happens and the status just remains "In Progress" can you, please confirm that you have enabled the "REPORT WORKFLOW STATUS = TRUE" in the Action Backend Tab.



Edit action - GitHub Workflow

Enable anyone in your organization to perform self-service actions.

Invocation type: Run GitHub Workflow

Choose an installation method

Install the cloud GitHub App here

Install the self-hosted GitHub App here

Choose workflow to trigger

Organization*: vvergara84

Repository*: exos_repo

Workflow file name*: github-actions-demo.yml

Report workflow status: Yes

Configure the invocation payload

The following values will be sent upon the action invocation. You can use JQ to reference the action trigger data by writing double brackets, for example: {{ .trigger.by.user.email }}. You can reference secrets using {{ .secrets["secret-name"] }}. Learn more

GitHub workflow inputs

```
{}
```

Can you please also provide the execution logs and an image of your configuration.

Thanks in advance, greetings

**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.

ANSWER

An Aggregation Property was added to the Service Blueprint to get the information

```
"open_prs_count":{
  "title":"Open PRs count",
  "icon":"DefaultProperty",
  "type":"number",
  "target":"githubPullRequest",
  "query":{
    "combinator":"and",
    "rules":[
      {
        "value":"open",
        "property":"status",
        "operator":"="
      }
    ]
  },
  "calculationSpec":{
    "func":"count",
    "calculationBy":"entities"
  },
  "pathFilter":[
    {
      "fromBlueprint":"githubPullRequest",
      "path":[
        "service"
      ]
    }
  ]
}
```

A Scorecard was created with the Rules for the PR Count

```
{
  "identifier":"pr_health",
  "title":"PR Health",
  "levels":[
    {
      "color":"paleBlue",
      "title":"Basic"
    },
    {
      "color":"bronze",
      "title":"Bronze"
    },
    {
      "color":"silver",
      "title":"Silver"
    },
    {
      "color":"gold",
      "title":"Gold"
    }
  ],
  "rules":[
    {
      "identifier":"fewer_than_15_open_p_rs",
      "title":"Fewer than 15 open PRs",
      "level":"Bronze",
      "query":{
        "combinator":"and",
        "conditions":[
          {
            "value":15,
            "operator":"<",
            "property":"open_prs_count"
          }
        ]
      }
    },
   {
     "identifier":"fewer_than_5_open_p_rs",
     "title":"Fewer than 5 open PRs",
     "level":"Gold",
     "query":{
       "combinator":"and",
       "conditions":[
         {
           "value":5,
           "operator":"<",
           "property":"open_prs_count"
         }
       ]
     }
   },
   {
     "identifier":"fewer_than_10_open_p_rs",
     "title":"Fewer than 10 open PRs",
     "level":"Silver",
     "query":{
       "combinator":"and",
       "conditions":[
         {
           "value":10,
           "operator":"<",
           "property":"open_prs_count"
         }
       ]
     }
   }
 ]
}
```

**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 **service** blueprint, you can use the same blueprint and entities that the GitHub app created for you
    - 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 or any other programming/scripting language that:
    - Queries the different service entities and frameworks using Port's API
    - Calculates the correct number of EOL packages for each service entity
    - Updates the value of the number of EOL packages property on each service entity using Port's API.

ANSWER

I added to the Service the Property Number of EOL Packages

```
"properties":{
  "number_of_eol_packages":{
    "type":"number",
    "title":"Number of EOL packages"
  }
}
```

I created a Framework Blueprint

```
{
  "identifier":"framework",
  "title":"Framework",
  "icon":"java",
  "schema":{
    "properties":{
      "state":{
        "type":"array",
        "title":"State",
        "default":[
          "ACTIVE"
        ],
        "items":{
          "enum":[
            "ACTIVE",
            "EOL"
          ],
          "enumColors":{
            "ACTIVE":"lightGray",
            "EOL":"lightGray"
          },
          "type":"string"
        }
      }
    },
    "required":[
      "state"
    ]
  },
  "mirrorProperties":{

  },
  "calculationProperties":{

  },
  "aggregationProperties":{

  },
  "relations":{

  }
}
```

I also added to the service the relation to Framework Blueprint

```
"relations":{
  "used_frameworks":{
    "title":"used_frameworks",
    "target":"framework",
    "required":false,
    "many":true
  }
}
```

Created an Action Self-Service to generate 5 Mock entities

```
{
  "identifier":"mock_eol_packages",
  "title":"Mock EOL Packages",
  "description":"Used to create Entities",
  "trigger":{
    "type":"self-service",
    "operation":"CREATE",
    "userInputs":{
      "properties":{

      },
      "required":[

      ],
      "order":[

      ]
    }
  },
  "invocationMethod":{
    "type":"UPSERT_ENTITY",
    "blueprintIdentifier":"framework",
    "mapping":{
      "identifier":"entity_5",
      "title":"Entity 5",
      "icon":"DefaultBlueprint",
      "properties":{
        "State":"EOL"
      }
    }
  },
  "requiredApproval":false,
  "allowAnyoneToViewRuns":true
}
```

Finally with the help of AI I created this Java Script:

```
import com.fasterxml.jackson.databind.ObjectMapper;
import com.fasterxml.jackson.databind.node.ArrayNode;
import com.fasterxml.jackson.databind.node.ObjectNode;

import okhttp3.*;

import java.io.IOException;
import java.util.*;

public class PortEolPackagesUpdater {
   private static final String PORT_BASE_URL = "https://api.getport.io/v1";
   private static final String PORT_CLIENT_ID = "RFbtPBJzvF1YGPBjHxIsIl00EidJkOsD";
   private static final String PORT_CLIENT_SECRET =
"3An8D3XE2wYtGVAXz3f3231YIOZN4DLGX4jOiI7hSFsukh7EeIVYMeKoMaqsAWSw";

   private static final String SERVICE_BLUEPRINT = "service";
   private static final String FRAMEWORK_BLUEPRINT = "framework";
   private static final String RELATION_NAME = "used_frameworks";
   private static final String EOL_PROPERTY = "state";
   private static final String EOL_VALUE = "EOL";
   private static final String TARGET_PROPERTY = "number_of_eol_packages";

   private final OkHttpClient client = new OkHttpClient();
   private final ObjectMapper mapper = new ObjectMapper();
   private String accessToken;

   public static void main(String[] args) {
     PortEolPackagesUpdater updater = new PortEolPackagesUpdater();

     try {
       updater.run();
     }
     catch (Exception e) {
       e.printStackTrace();
       System.exit(1);
     }
   }

         private void run() throws IOException {
     accessToken = getPortToken();

     System.out.println("Access Token: " + accessToken);

     List<Map<String, Object>> services = listEntities(SERVICE_BLUEPRINT);

     System.out.println("Service Entities List: " + services.toString());

     for (Map<String, Object> service : services) {
       String serviceId = (String) service.get("identifier");

       System.out.println("ServiceId: " + serviceId);
```

```
        List<String> frameworkIds = getRelationIdentifiers(service, RELATION_NAME);

        System.out.println("Frameworks associated to the Service: " + frameworkIds.toString());

        int eolCount = 0;

        for (String frameworkId : frameworkIds) {
          System.out.println("Getting Entity Id: " + frameworkId);

          Map<String, Object> framework = getEntity(FRAMEWORK_BLUEPRINT, frameworkId);

          ArrayList state = (ArrayList) (((Map) ((Map) framework.get("entity")).get("properties")).get(EOL_PROPERTY));

          System.out.println("a: " + state.get(0));

          if (EOL_VALUE.equals(state.get(0).toString())) {
            eolCount++;
          }
        }
        System.out.println("Number of EOL Entities: " + eolCount);

        updateServiceEolCount(serviceId, eolCount);

        System.out.printf("Updated service %s with %d EOL packages%n", serviceId, eolCount);
      }
    }

    private String getPortToken() throws IOException {
          System.out.println("Obtaining Access Token");
      String url = PORT_BASE_URL + "/auth/access_token";

      ObjectNode payload = mapper.createObjectNode().put("clientId", PORT_CLIENT_ID).put("clientSecret", PORT_CLIENT_SECRET);

      RequestBody body = RequestBody.create(payload.toString(), MediaType.get("application/json"));

      Request request = new Request.Builder().url(url).post(body).build();

      try (Response response = client.newCall(request).execute()) {
        if (!response.isSuccessful()) {
          throw new IOException("Auth failed: " + response);
        }
        ObjectNode json = (ObjectNode) mapper.readTree(response.body().string());

        return json.get("accessToken").asText();
      }
    }

    @SuppressWarnings("unchecked")
    private List<Map<String, Object>> listEntities(String blueprint) throws IOException {
          System.out.println("Obtaining Service Entities List");

      List<Map<String, Object>> entities = new ArrayList<>();
      String cursor = null;

      do {
        HttpUrl.Builder urlBuilder = HttpUrl.parse(PORT_BASE_URL + "/blueprints/" + blueprint + "/entities").newBuilder();
        if (cursor != null) {
          urlBuilder.addQueryParameter("cursor", cursor);
        }
```

```
        Request request = new Request.Builder().url(urlBuilder.build()).addHeader("Authorization", "Bearer " + accessToken).build();

        try (Response response = client.newCall(request).execute()) {
          if (!response.isSuccessful()) {
                throw new IOException("Failed to list entities: " + response);
          }
          ObjectNode json = (ObjectNode) mapper.readTree(response.body().string());

          ArrayNode entitiesArray = (ArrayNode) json.get("entities");

          for (int i = 0; i < entitiesArray.size(); i++) {
            entities.add(mapper.convertValue(entitiesArray.get(i), Map.class));
          }

          cursor = json.has("nextCursor") ? json.get("nextCursor").asText() : null;
        }
      } while (cursor != null);

      return entities;
    }

    @SuppressWarnings("unchecked")
    private List<String> getRelationIdentifiers(Map<String, Object> entity, String relationName) {
          System.out.println("RelationName: " + relationName);

          List<Object> relations = null;

          Object relationsObj = entity.getOrDefault("relations", Collections.emptyMap());

          if (relationsObj instanceof Map) {
            Map<String, Object> relationsMap = (Map<String, Object>) relationsObj;


            Object relationValue = relationsMap.get(relationName);
            relations = (relationValue instanceof List) ? (List<Object>) relationValue :
Collections.emptyList();
          }
          else {
            relations = Collections.emptyList();
          }


      if (relations == null) {
          return Collections.emptyList();
      }
      return
relations.stream().map(Object::toString).collect(java.util.stream.Collectors.toList());
   }


   @SuppressWarnings("unchecked")
   private Map<String, Object> getEntity(String blueprint, String identifier) throws
IOException {
          System.out.println("Getting Framework Entities");


      String url = PORT_BASE_URL + "/blueprints/" + blueprint + "/entities/" + identifier;


      Request request = new Request.Builder().url(url).addHeader("Authorization", "Bearer " +
accessToken).build();


      System.out.println("Request: " + request);


      try (Response response = client.newCall(request).execute()) {
        if (!response.isSuccessful()) {
          throw new IOException("Failed to get entity: " + response);
        }
        Map<String, Object> map = mapper.readValue(response.body().string(), Map.class);
        System.out.println("Response: " + map);

        return map;
      }
    }

    private void updateServiceEolCount(String serviceId, int eolCount) throws IOException {
          System.out.println("Update EOL Count");

      String url = PORT_BASE_URL + "/blueprints/" + SERVICE_BLUEPRINT + "/entities/" +
serviceId;

      ObjectNode payload = mapper.createObjectNode().set("properties",
mapper.createObjectNode().put(TARGET_PROPERTY, eolCount));

      RequestBody body = RequestBody.create(payload.toString(),
MediaType.get("application/json"));

      Request request = new
Request.Builder().url(url).patch(body).addHeader("Authorization", "Bearer " +
accessToken).addHeader("Content-Type", "application/json").build();

      try (Response response = client.newCall(request).execute()) {
        if (!response.isSuccessful()) {
          throw new IOException("Failed to update service " + serviceId + ": " + response);
        }
      }
    }
}
```