如何使用 Argo Event & Workflow 快速建置自定義的工作流程

Wei
2022.04.29

LINE

Wei

LINE TW SRE

喜歡寫 Go

喜歡玩 Kubernetes

有時候會貢獻一些開源專案

每天一杯咖啡

Contents

01 Infrastructure in LINE

02 TW SRE Team & Products

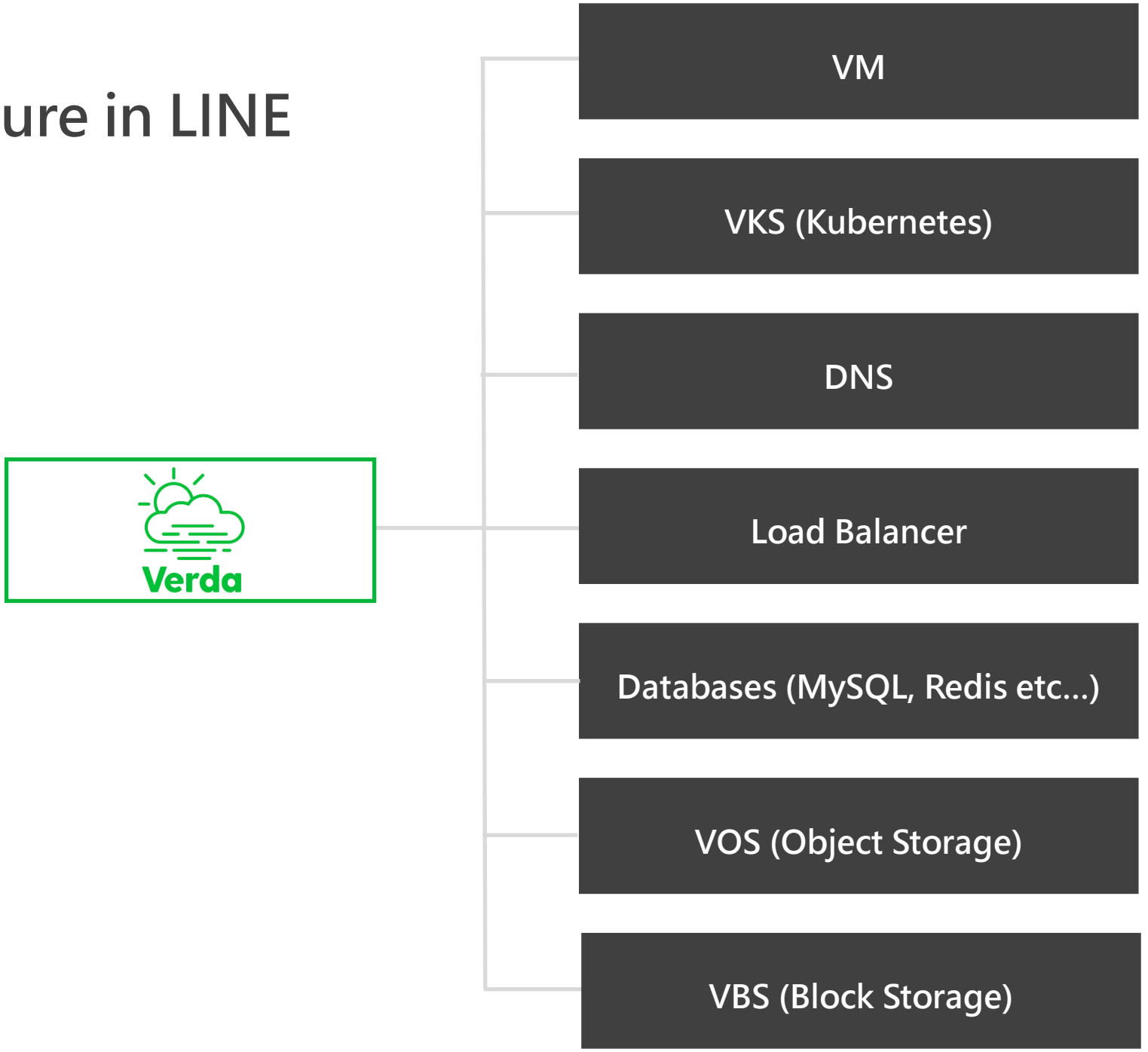
03 Taiwan Observability Platform

04 Argo Workflows

05 LHCI Farm

06 Argo Events

Infrastructure in LINE



TW SRE Team

日常

Observability

- K8s 應用監控、微服務分布式跟蹤、日誌匯聚搜尋
- 提供整個台灣辦公室各應用開發團隊的全域監控
- 目前已經使用的OSS專案包含
  - Grafana
  - Grafana Loki
  - Prometheus
  - Grafana Tempo
  - OpenTelemetry/Jaeger(legacy)
- 告警系統整合(Slack)

TW SRE Team

日常

Serverless

- 提供服務系統，自動化配置應用的路由、升級策略、自動擴縮容等功能
- 目前已經使用的OSS專案包含
  - Rancher
  - Knative Serving
  - Argo CD / Argo Workflows

TW SRE Team

日常

Operations Support

- 應用開發團隊 K8s 維運支援
- 協助小型開發團隊維運

TW SRE Team

日常

CI/CD

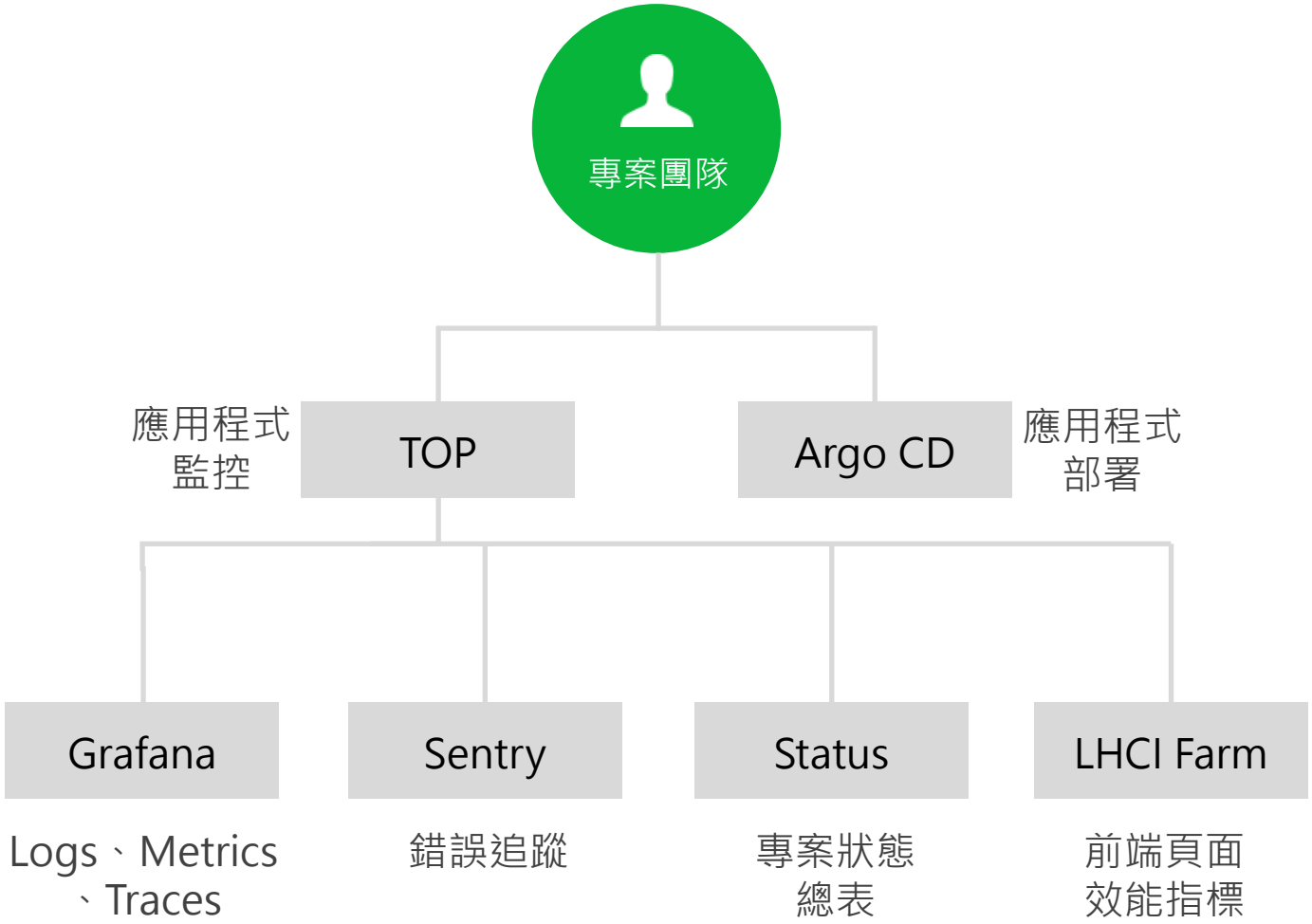
- 代管 Argo CD 推行 GitOps
- 提供訂製 Drone plugin 支援特殊CI需求
- Lighthouse 整合前端頁面效能指標
- 目前已經使用的OSS專案包含
  - Argo CD
  - Lighthouse CI

Taiwan Observability Platform

數據觀察是為洞察先機， TOP 讓數據分析變成藝術

TW SRE Team

產品 – Taiwan Observability Platform



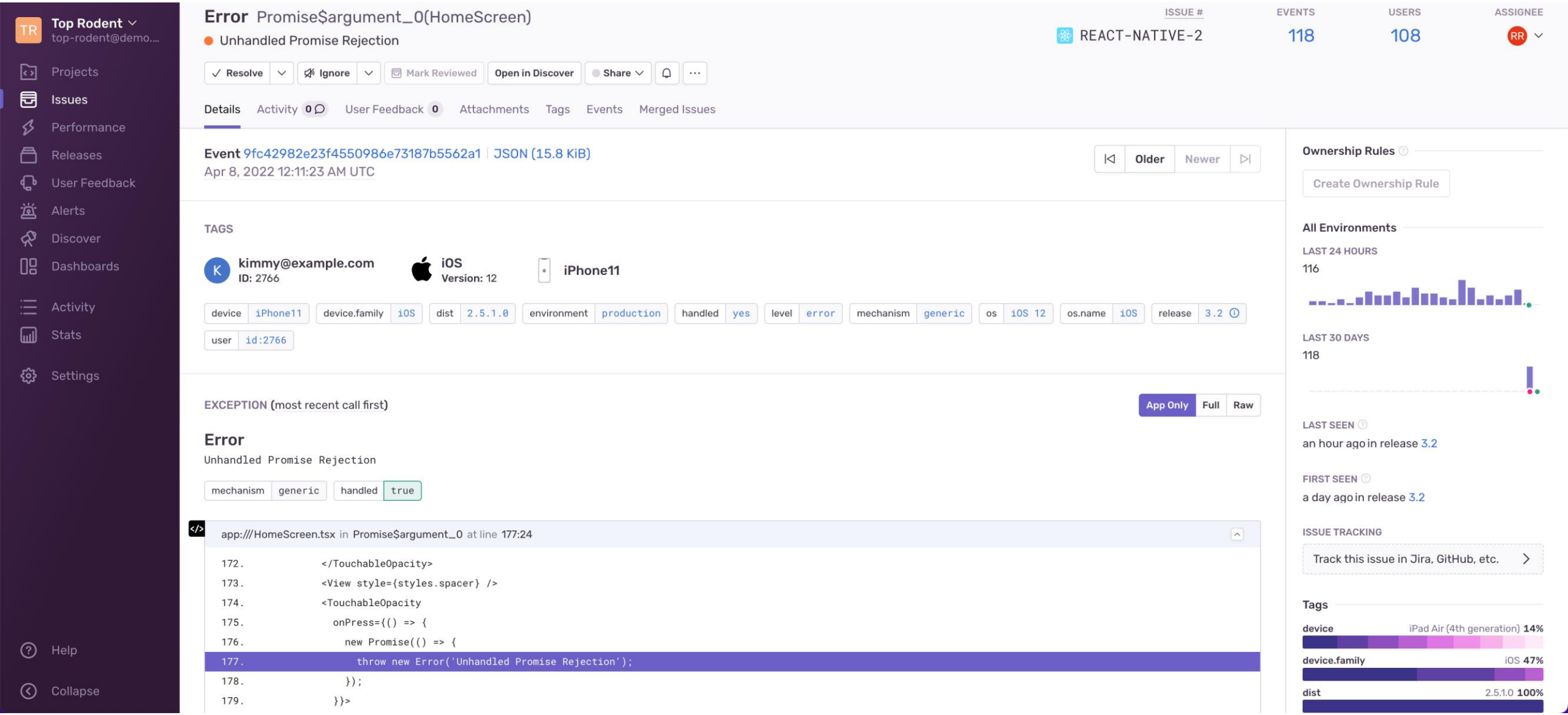
Argo Workflows 應用

Self-hosted Sentry

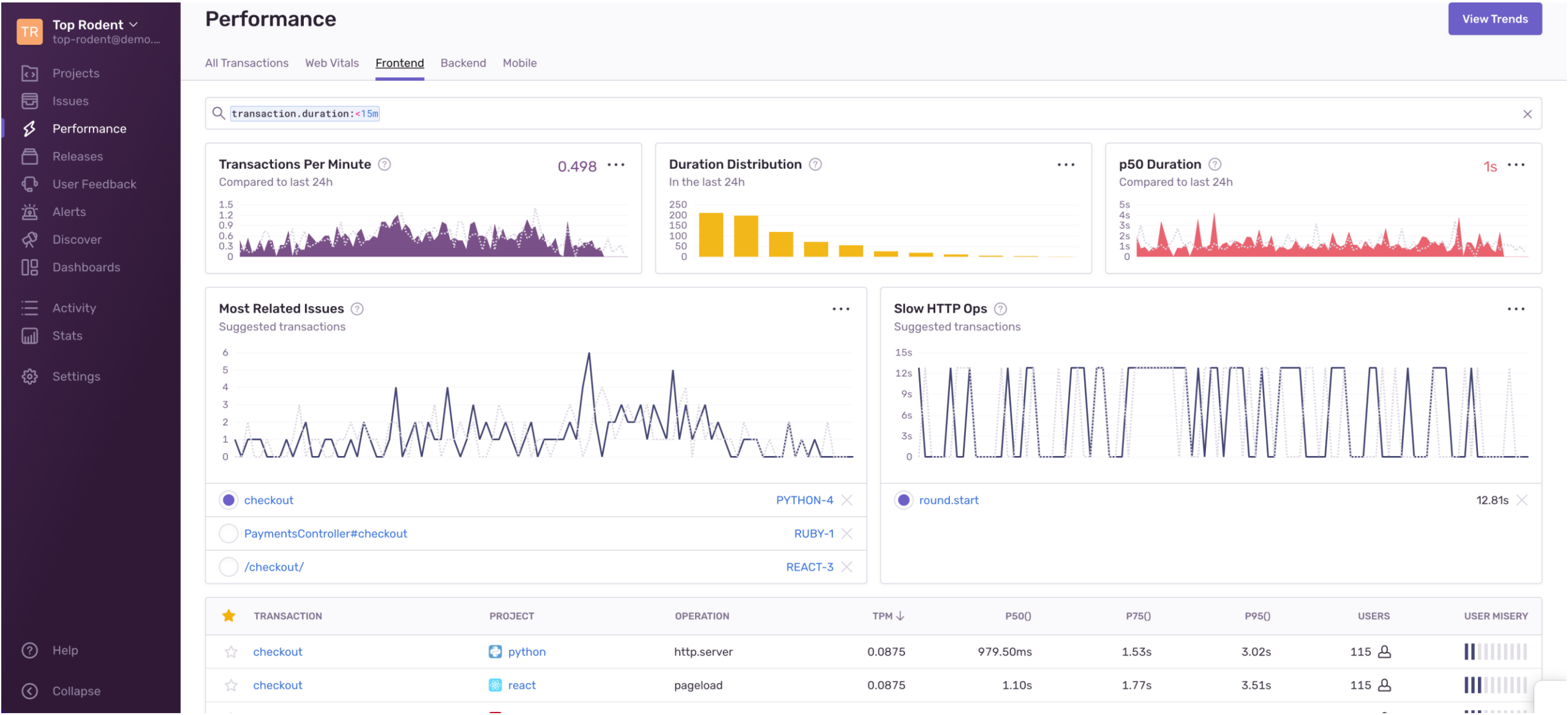
背景

什麼是 Sentry ?

遇到什麼問題



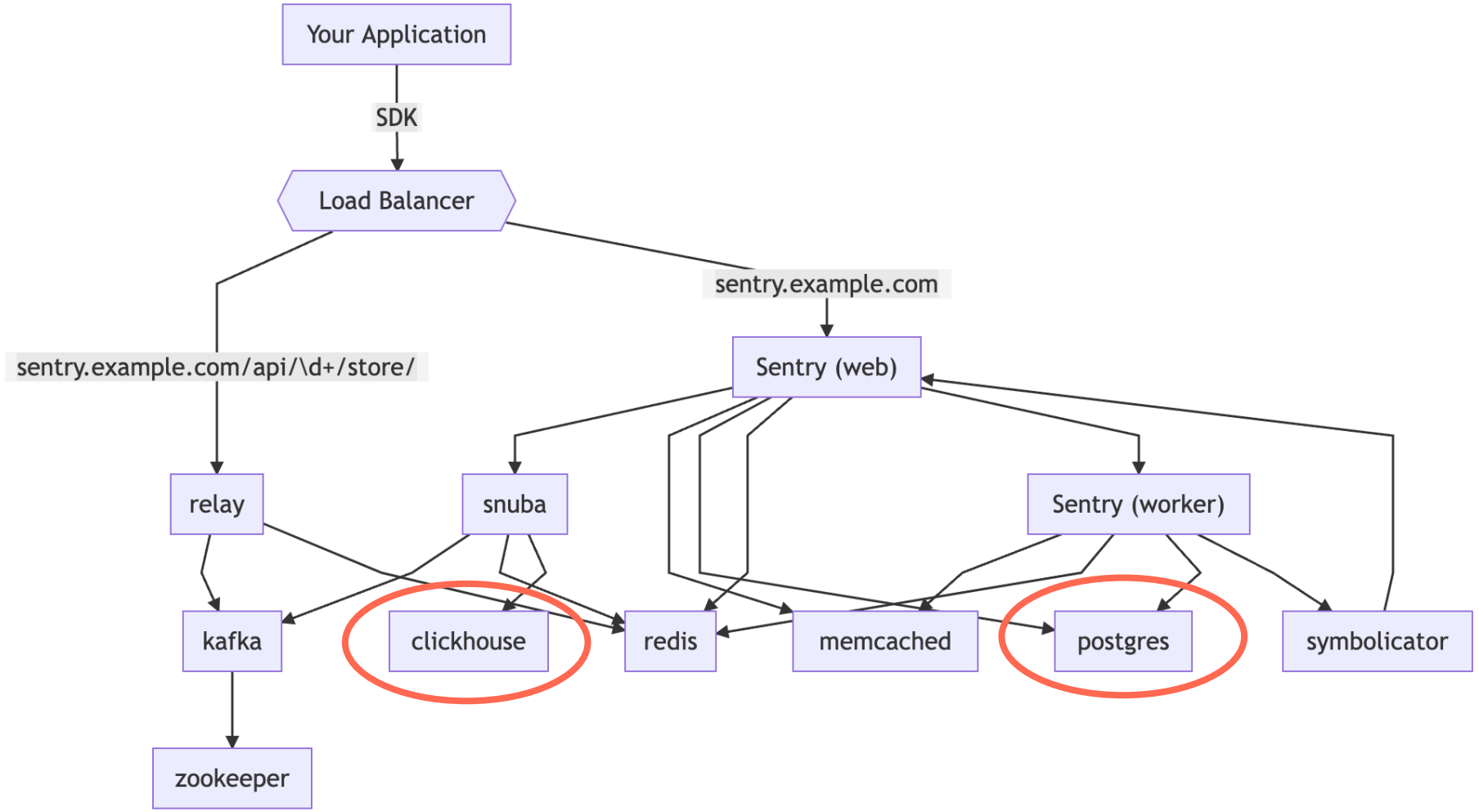
※Source from：https://try.sentry-demo.com/



※Source from：https://try.sentry-demo.com/

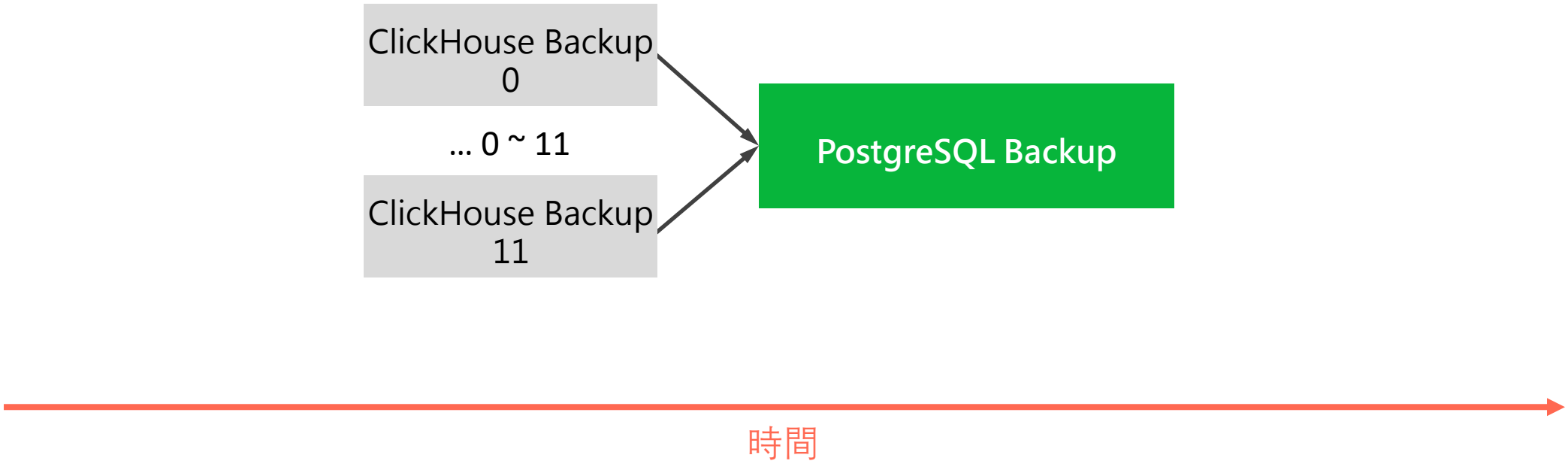
為什麼需要 Argo Workflows ?

從自建 Sentry 開始



※Source from：https://develop.sentry.dev/architecture/

關於Sentry資料庫的備份



如何處理

Kubernetes Cronjob

```yaml
apiVersion: batch/v1beta1
kind: CronJob
metadata:
  name: sentry-db-backup
spec:
  ...
  jobTemplate:
    spec:
      template:
        spec:
          ...
          containers:
          - name: s3cmd
            image: d3fk/s3cmd:stable
            ...
            command:
            - sh
            - -c
            - |
              #!/bin/bash
              ...
```

```
#!/bin/bash
apk add --no-cache curl jq pigz postgresql-client
curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
chmod a+x ./kubectl
mv ./kubectl /usr/local/bin/

printf '\nRun clickhouse-backup create_remote...\n'
kubectl -n sentry exec chi-snuba-snuba-0-0-0 -c clickhouse-backup -- clickhouse-backup create_remote $(date +"%Y-%m-%d") & \
kubectl -n sentry exec chi-snuba-snuba-1-0-0 -c clickhouse-backup -- clickhouse-backup create_remote $(date +"%Y-%m-%d") & \
kubectl -n sentry exec chi-snuba-snuba-2-0-0 -c clickhouse-backup -- clickhouse-backup create_remote $(date +"%Y-%m-%d") & \
kubectl -n sentry exec chi-snuba-snuba-3-0-0 -c clickhouse-backup -- clickhouse-backup create_remote $(date +"%Y-%m-%d") & \
kubectl -n sentry exec chi-snuba-snuba-4-0-0 -c clickhouse-backup -- clickhouse-backup create_remote $(date +"%Y-%m-%d") & \
kubectl -n sentry exec chi-snuba-snuba-5-0-0 -c clickhouse-backup -- clickhouse-backup create_remote $(date +"%Y-%m-%d") & \
kubectl -n sentry exec chi-snuba-snuba-6-0-0 -c clickhouse-backup -- clickhouse-backup create_remote $(date +"%Y-%m-%d") & \
kubectl -n sentry exec chi-snuba-snuba-7-0-0 -c clickhouse-backup -- clickhouse-backup create_remote $(date +"%Y-%m-%d") & \
kubectl -n sentry exec chi-snuba-snuba-8-0-0 -c clickhouse-backup -- clickhouse-backup create_remote $(date +"%Y-%m-%d") & \
kubectl -n sentry exec chi-snuba-snuba-9-0-0 -c clickhouse-backup -- clickhouse-backup create_remote $(date +"%Y-%m-%d") & \
kubectl -n sentry exec chi-snuba-snuba-10-0-0 -c clickhouse-backup -- clickhouse-backup create_remote $(date +"%Y-%m-%d") & \
kubectl -n sentry exec chi-snuba-snuba-11-0-0 -c clickhouse-backup -- clickhouse-backup create_remote $(date +"%Y-%m-%d") & \
wait

printf '\nCreate /root/.s3cfg\n'
tee -a /root/.s3cfg << EOF
[default]
host_base = xxxxxx
host_bucket = twsre-brown-db-backup
access_key = xxxxx
secret_key = xxxxxx
enable_multipart = True
multipart_chunk_size_mb = 50
multipart_max_chunks = 10000
EOF
```

優缺點

優

- 開發快速
- 很容易理解
- 利用Kubernetes原生的功能不用額外裝東西

缺

- 腳本到處複製貼上
- Job沒辦法單獨重用
- 腳本中途失敗不好除錯

Argo Workflows

建立可重用的 Cronjob，並滿足複雜 Job 依賴關係

Argo Workflows

什麼是 Argo Workflows ?

Argo Workflows 是一個開源容器原生工作流引擎，用於在 Kubernetes 上編排並行作業。

Argo Workflows 實現為 Kubernetes CRD (自定義資源定義) 。

- 定義工作流中的每個步驟都是一個容器的工作流。
- 將多步驟工作流建模為一系列任務或使用有向無環圖 (DAG) 捕獲任務之間的依賴關係。
- 使用 Kubernetes 上的 Argo Workflows 在很短的時間內輕鬆運行機器學習或數據處理的計算密集型作業。
- 在 Kubernetes 上本地運行 CI/CD 管道，無需配置複雜的軟件開發產品。

※Source from：https://argoproj.github.io/argo-workflows/

Core Concepts

Workflow

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  name: hello-world
spec:
  entrypoint: whalesay
  templates: ← Steps
  - name: whalesay
    container:
      image: docker/whalesay
      command:
      - cowsay
      args:
      - hello world
```

```yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: hello-world
spec:
  template:
    spec:
      restartPolicy: Never
      containers:
      - name: whalesay
        image: docker/whalesay
        command:
        - cowsay
        args:
        - hello world
```


※Source from：https://argoproj.github.io/argo-workflows/workflow-concepts/

Template types

container, script

```yaml
templates:
- name: whalesay
  container:
    image: docker/whalesay
    command:
    - cowsay
    args:
    - hello world
```

```yaml
templates:
- name: gen-random-int
  script:
    image: python:alpine3.6
    command:
    - python
    source: |
      import random
      i = random.randint(1, 100)
      print(i)
```


※Source from : https://argoproj.github.io/argo-workflows/workflow-concepts/

Template types

resource, delay

```
templates:
- name: k8s-owner-reference
  resource:
    action: create
    manifest: |
      apiVersion: v1
      kind: ConfigMap
      metadata:
        generateName: owned-eg-
      data:
        some: value
```

```
templates:
- name: delay
  suspend:
    duration: 20s
```


※Source from：https://argoproj.github.io/argo-workflows/workflow-concepts/

Template types

http

```yaml
templates:
- name: http
  http:
    timeoutSeconds: 20
    url: https://www.google.com
    method: "GET"
    headers:
    - name: "x-header-name"
      value: "test-value"
    body: "test body"
    successCondition: "response.body contains \"google\""
    # Available variables:
    #  request.body: string, the response body
    #  request.headers: map[string][]string, the response headers
    #  response.url: string, the request url
    #  response.method: string, the request method
    #  response.statusCode: int, the response status code
    #  response.body: string, the response body
    #  response.headers: map[string][]string, the response headers
```

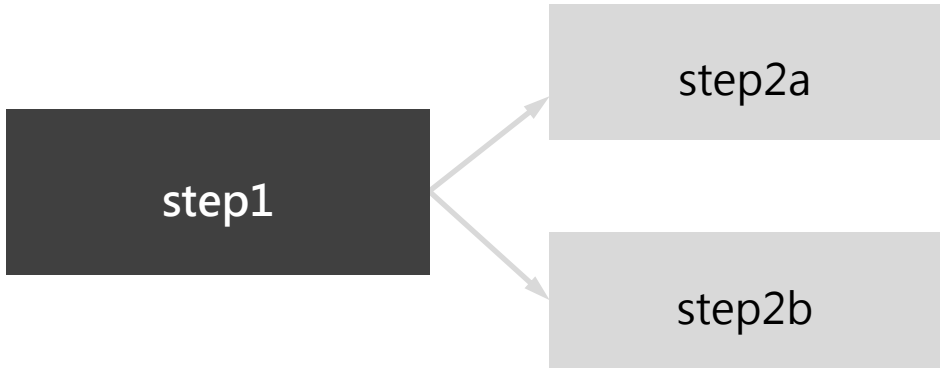

※Source from：https://argoproj.github.io/argo-workflows/http-template/

Template types

steps

```yaml
templates:
- name: hello-hello-hello
  steps:
  - - name: step1
      template: prepare-data
  - - name: step2a
      template: run-data-first-half
    - name: step2b
      template: run-data-second-half
```

引用其他 step



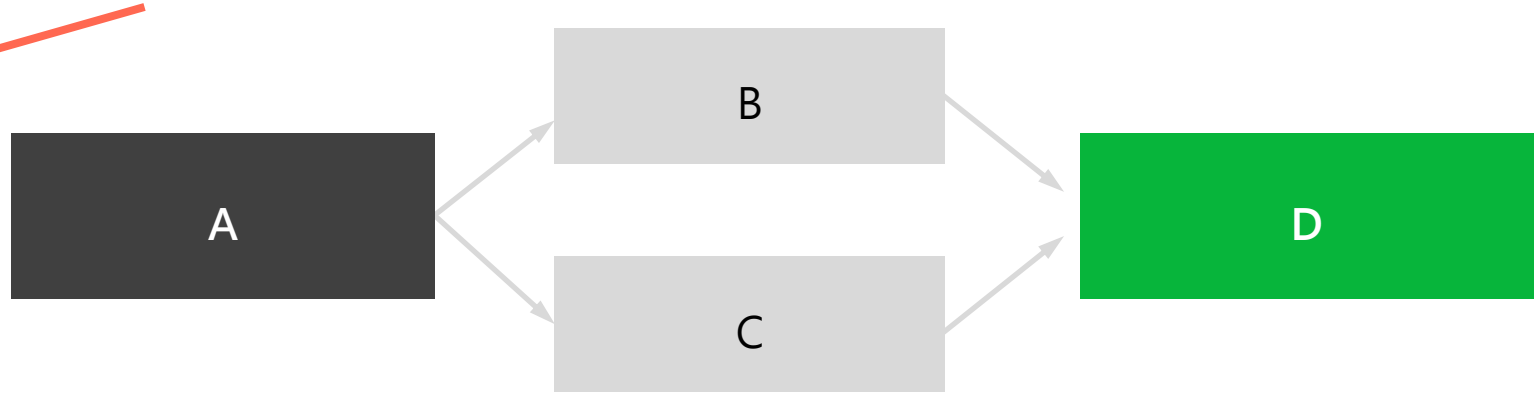
※Source from：https://argoproj.github.io/argo-workflows/workflow-concepts/

Template types

dag

```yaml
templates:
- name: diamond
  dag:
    tasks:
    - name: A
      template: echo
    - name: B
      dependencies:
      - A
      template: echo
    - name: C
      dependencies:
      - A
      template: echo
    - name: D
      dependencies:
      - B
      - C
      template: echo
```

引用其他 step



※Source from：https://argoproj.github.io/argo-workflows/workflow-concepts/

Parameters

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: hello-world-parameters-
spec:
  entrypoint: whalesay
  arguments:
    parameters:
    - name: message
      value: hello world
  templates:
  - name: whalesay
    inputs:
      parameters:
      - name: message
    container:
      image: docker/whalesay
      command:
      - cowsay
      args:
      - "{{inputs.parameters.message}}"
```

以 “hello world” 作為 message參數

參數聲明

模板語言引用參數

※Source from：https://github.com/argoproj/argo-workflows/blob/master/examples/README.md#parameters

實戰

WorkflowTemplates & ClusterWorkflowTemplate

Loops

Web UI

```yaml
apiVersion: argoproj.io/v1alpha1
kind: ClusterWorkflowTemplate
metadata:
  name: clickhouse
spec:
  templates:
  - name: backup
    inputs:
      parameters:
      - name: clickhouse_pod
    script:
      image: alpine:3.15.0
      imagePullPolicy: IfNotPresent
      command:
      - sh
      source: |
        apk add --no-cache curl
        curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
        chmod a+x ./kubectl
        mv ./kubectl /usr/local/bin/

        printf '\nRun clickhouse-backup delete remote...\n'
        kubectl -n sentry exec {{ inputs.parameters.clickhouse_pod }} -c clickhouse-backup -- clickhouse-backup delete remote $(date +"%Y-%m-%d")

        printf '\nRun clickhouse-backup create_remote...\n'
        kubectl -n sentry exec {{ inputs.parameters.clickhouse_pod }} -c clickhouse-backup -- clickhouse-backup create_remote $(date +"%Y-%m-%d")

        printf '\nRun clickhouse-backup delete local...\n'
        kubectl -n sentry exec {{ inputs.parameters.clickhouse_pod }} -c clickhouse-backup -- clickhouse-backup delete local $(date +"%Y-%m-%d")

        printf '\nDone'
```

```yaml
apiVersion: argoproj.io/v1alpha1
kind: CronWorkflow
metadata:
  name: sentry-db-backup
spec:
  schedule: 0 4 * * *
  timezone: Asia/Taipei
  workflowSpec:
    serviceAccountName: workflow
    entrypoint: main
    templates:
    - name: main
      steps:
      - - name: clickhouse-backup
          templateRef:
            clusterScope: true
            name: clickhouse
            template: backup
          arguments:
            parameters:
            - name: clickhouse_pod
              value: "{{ item }}"
          withItems:
          - chi-snuba-snuba-0-0-0
```

指定每天運行的時間跟時區

引用Workflow Template

```
    - - name: clickhouse-backup
        templateRef:
          clusterScope: true
          name: clickhouse
          template: backup
        arguments:
          parameters:
          - name: clickhouse_pod
            value: "{{ item }}"
        withItems:
        - chi-snuba-snuba-0-0-0
        - chi-snuba-snuba-1-0-0
        - chi-snuba-snuba-2-0-0
        - chi-snuba-snuba-3-0-0
        - chi-snuba-snuba-4-0-0
        - chi-snuba-snuba-5-0-0
        - chi-snuba-snuba-6-0-0
        - chi-snuba-snuba-7-0-0
        - chi-snuba-snuba-8-0-0
        - chi-snuba-snuba-9-0-0
        - chi-snuba-snuba-10-0-0
        - chi-snuba-snuba-11-0-0
    - - name: postgres-backup
        templateRef:
          clusterScope: true
```

自動展開帶入參數到
每個item展開成steps

Cron Workflows / sentry

CRON WORKFLOWS

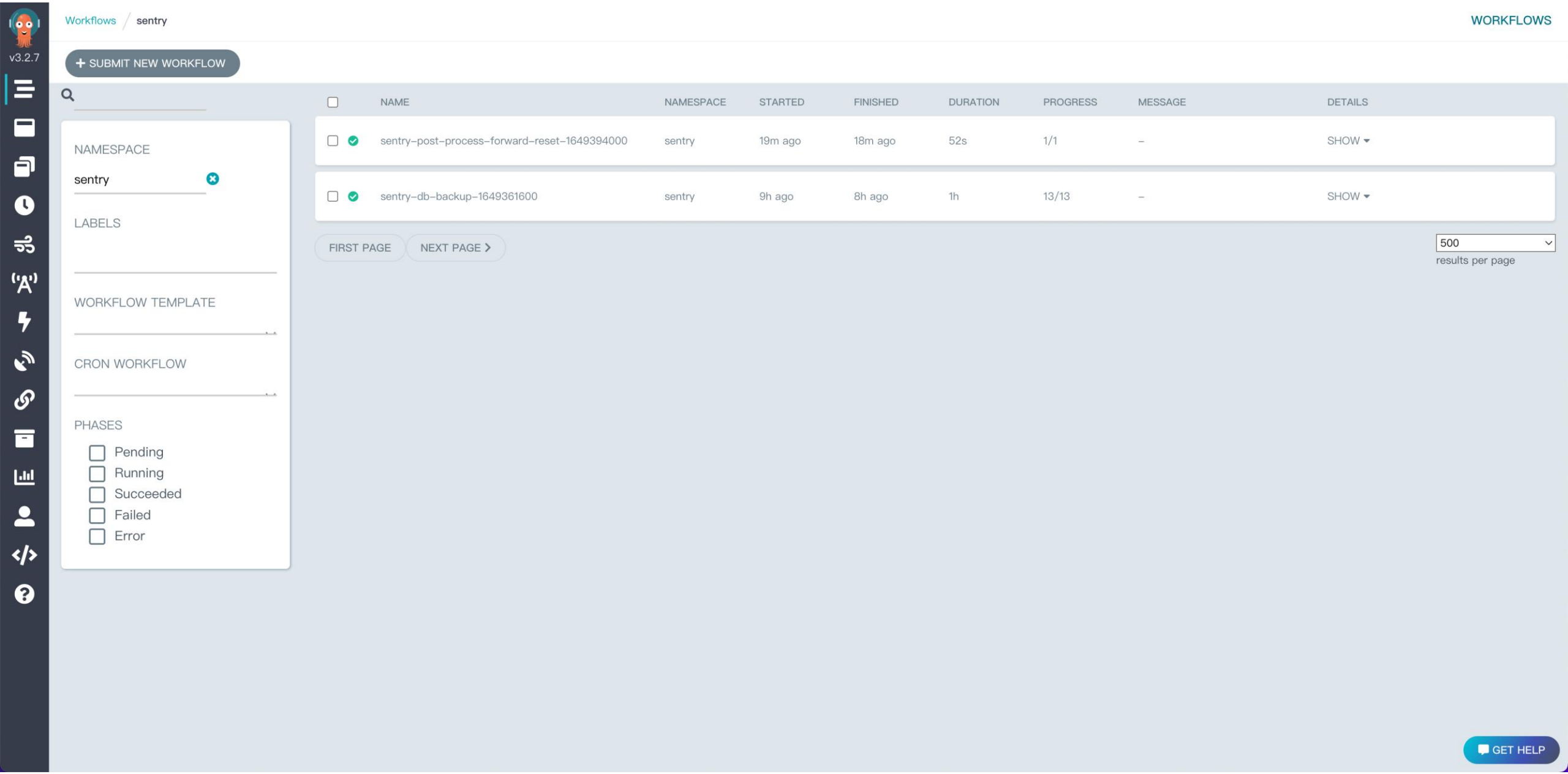
+ CREATE NEW CRON WORKFLOW

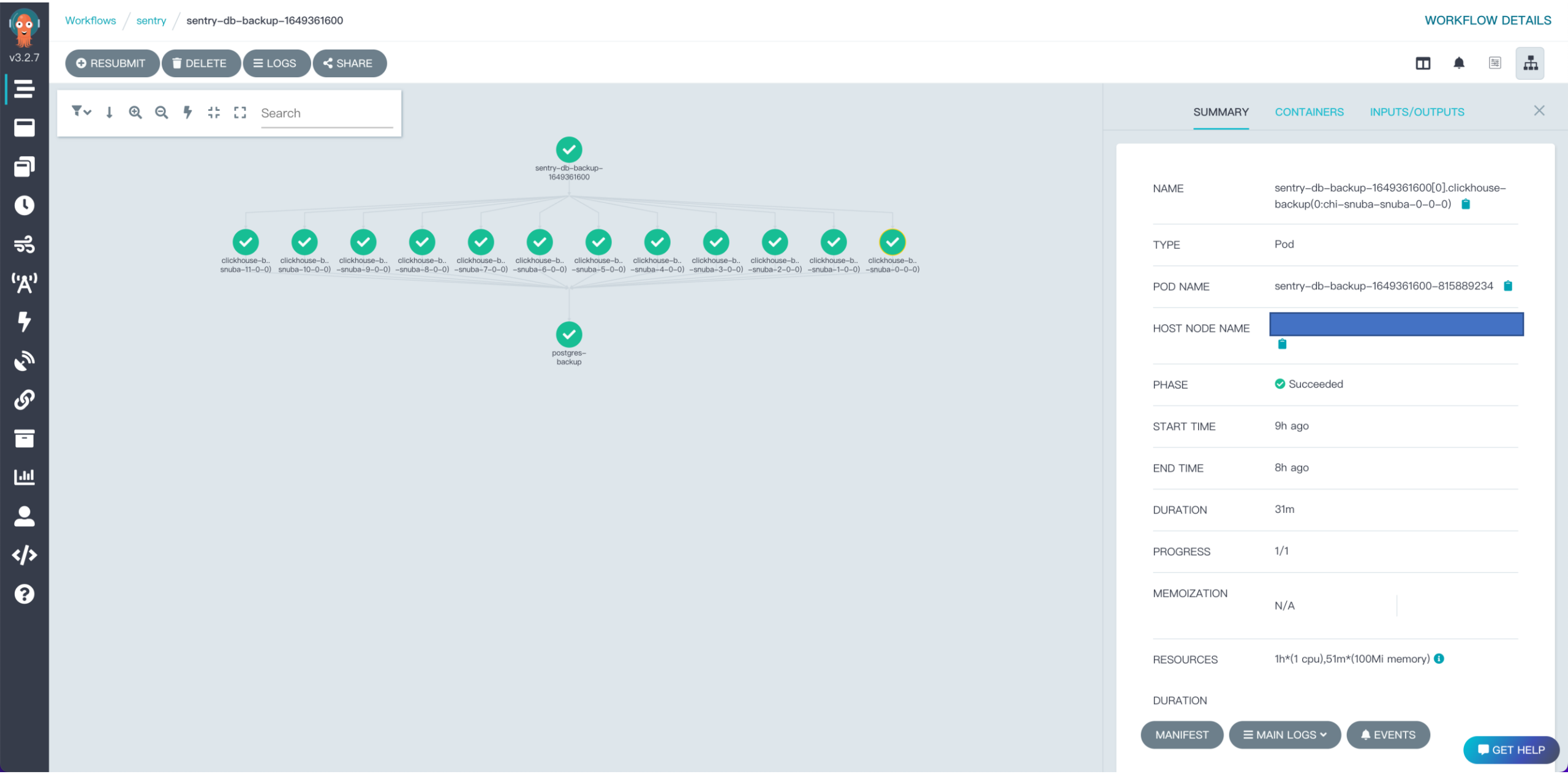
sentry

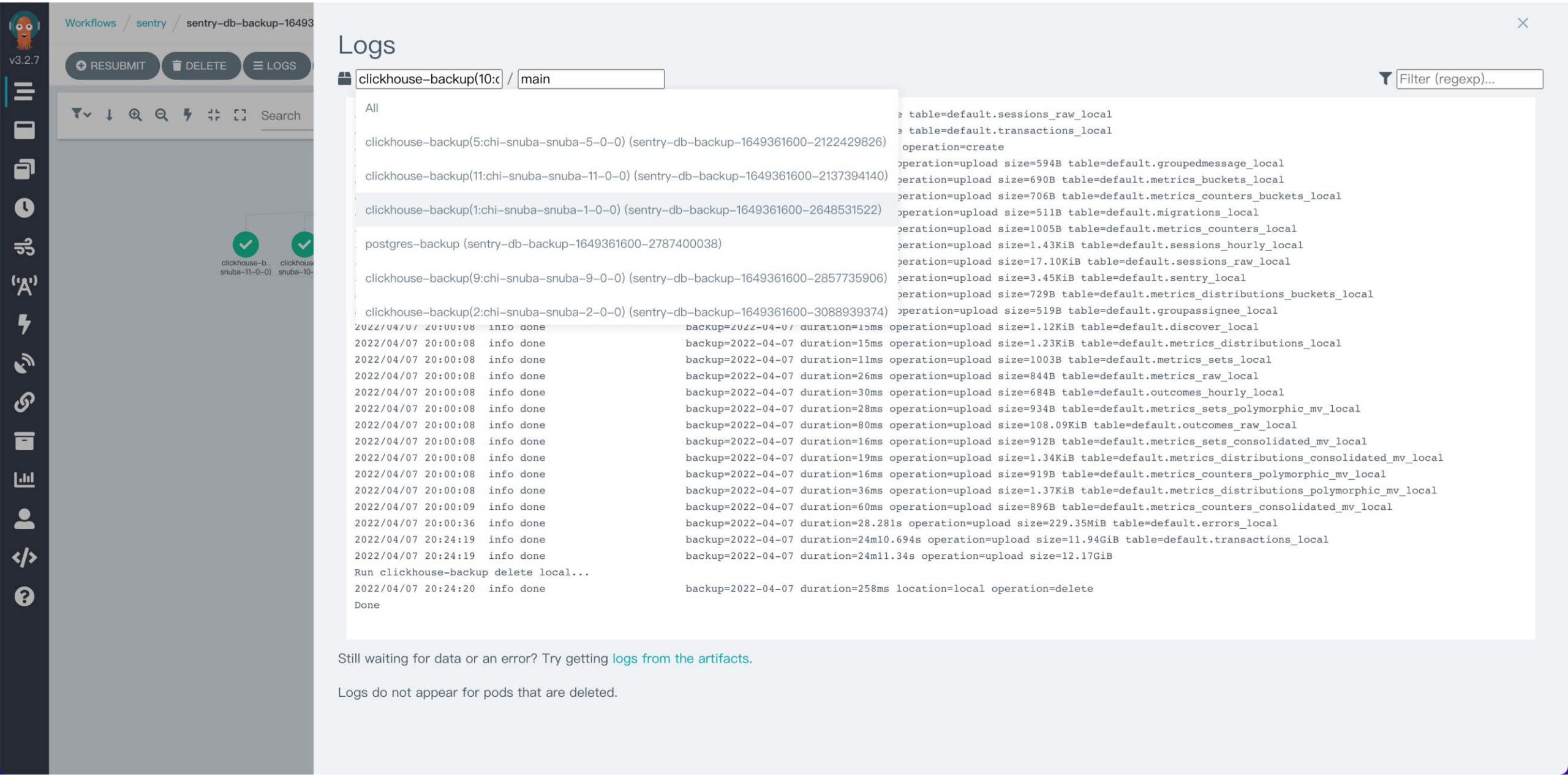
NAME	NAMESPACE	SCHEDULE		CREATED	NEXT RUN
sentry-db-backup	sentry	0 4 * * *	At 04:00 AM	60d ago	in 14h
sentry-post-process-forward-reset	sentry	0 * * * *	Every hour	27d ago	in 41m

Cron workflows are workflows that run on a preset schedule. Next scheduled run assumes workflow-controller is in UTC. You can find manifests in the examples or templates in Workflow Template Catalog. Learn more.

GET HELP


v3.2.7
Workflows / sentry
WORKFLOWS
+ SUBMIT NEW WORKFLOW
NAMESPACE
sentry
LABELS
WORKFLOW TEMPLATE
CRON WORKFLOW
PHASES
Pending
Running
Succeeded
Failed
Error
NAME
NAMESPACE
STARTED
FINISHED
DURATION
PROGRESS
MESSAGE
DETAILS
sentry-post-process-forward-reset-1649394000
sentry
19m ago
18m ago
52s
1/1
–
SHOW
sentry-db-backup-1649361600
sentry
9h ago
8h ago
1h
13/13
–
SHOW
FIRST PAGE
NEXT PAGE >
500
results per page
GET HELP


v3.2.7
Workflows / sentry / sentry-db-backup-1649361600
WORKFLOW DETAILS
RESUBMIT
DELETE
LOGS
SHARE
Search
sentry-db-backup-1649361600
clickhouse-b..snuba-11-0-0)
clickhouse-b..snuba-10-0-0)
clickhouse-b..-snuba-9-0-0)
clickhouse-b..-snuba-8-0-0)
clickhouse-b..-snuba-7-0-0)
clickhouse-b..-snuba-6-0-0)
clickhouse-b..-snuba-5-0-0)
clickhouse-b..-snuba-4-0-0)
clickhouse-b..-snuba-3-0-0)
clickhouse-b..-snuba-2-0-0)
clickhouse-b..-snuba-1-0-0)
clickhouse-b..-snuba-0-0-0)
postgres-backup
SUMMARY
CONTAINERS
INPUTS/OUTPUTS
NAME
sentry-db-backup-1649361600[0].clickhouse-backup(0:chi-snuba-snuba-0-0-0)
TYPE
Pod
POD NAME
sentry-db-backup-1649361600-815889234
HOST NODE NAME
PHASE
Succeeded
START TIME
9h ago
END TIME
8h ago
DURATION
31m
PROGRESS
1/1
MEMOIZATION
N/A
RESOURCES
1h*(1 cpu),51m*(100Mi memory)
DURATION
MANIFEST
MAIN LOGS
EVENTS
GET HELP


v3.2.7
Workflows / sentry / sentry-db-backup-16493
RESUBMIT
DELETE
LOGS
Search
Logs
clickhouse-backup(10:c
main
Filter (regexp)...
All
clickhouse-backup(5:chi-snuba-snuba-5-0-0) (sentry-db-backup-1649361600-2122429826)
clickhouse-backup(11:chi-snuba-snuba-11-0-0) (sentry-db-backup-1649361600-2137394140)
clickhouse-backup(1:chi-snuba-snuba-1-0-0) (sentry-db-backup-1649361600-2648531522)
postgres-backup (sentry-db-backup-1649361600-2787400038)
clickhouse-backup(9:chi-snuba-snuba-9-0-0) (sentry-db-backup-1649361600-2857735906)
clickhouse-backup(2:chi-snuba-snuba-2-0-0) (sentry-db-backup-1649361600-3088939374)
e table=default.sessions_raw_local
e table=default.transactions_local
operation=create
peration=upload size=594B table=default.groupedmessage_local
peration=upload size=690B table=default.metrics_buckets_local
peration=upload size=706B table=default.metrics_counters_buckets_local
peration=upload size=511B table=default.migrations_local
peration=upload size=1005B table=default.metrics_counters_local
peration=upload size=1.43KiB table=default.sessions_hourly_local
peration=upload size=17.10KiB table=default.sessions_raw_local
peration=upload size=3.45KiB table=default.sentry_local
peration=upload size=729B table=default.metrics_distributions_buckets_local
peration=upload size=519B table=default.groupassignee_local
2022/04/07 20:00:08 info done backup=2022-04-07 duration=15ms operation=upload size=1.12KiB table=default.discover_local
2022/04/07 20:00:08 info done backup=2022-04-07 duration=15ms operation=upload size=1.23KiB table=default.metrics_distributions_local
2022/04/07 20:00:08 info done backup=2022-04-07 duration=11ms operation=upload size=1003B table=default.metrics_sets_local
2022/04/07 20:00:08 info done backup=2022-04-07 duration=26ms operation=upload size=844B table=default.metrics_raw_local
2022/04/07 20:00:08 info done backup=2022-04-07 duration=30ms operation=upload size=684B table=default.outcomes_hourly_local
2022/04/07 20:00:08 info done backup=2022-04-07 duration=28ms operation=upload size=934B table=default.metrics_sets_polymorphic_mv_local
2022/04/07 20:00:08 info done backup=2022-04-07 duration=80ms operation=upload size=108.09KiB table=default.outcomes_raw_local
2022/04/07 20:00:08 info done backup=2022-04-07 duration=16ms operation=upload size=912B table=default.metrics_sets_consolidated_mv_local
2022/04/07 20:00:08 info done backup=2022-04-07 duration=19ms operation=upload size=1.34KiB table=default.metrics_distributions_consolidated_mv_local
2022/04/07 20:00:08 info done backup=2022-04-07 duration=16ms operation=upload size=919B table=default.metrics_counters_polymorphic_mv_local
2022/04/07 20:00:08 info done backup=2022-04-07 duration=36ms operation=upload size=1.37KiB table=default.metrics_distributions_polymorphic_mv_local
2022/04/07 20:00:09 info done backup=2022-04-07 duration=60ms operation=upload size=896B table=default.metrics_counters_consolidated_mv_local
2022/04/07 20:00:36 info done backup=2022-04-07 duration=28.281s operation=upload size=229.35MiB table=default.errors_local
2022/04/07 20:24:19 info done backup=2022-04-07 duration=24m10.694s operation=upload size=11.94GiB table=default.transactions_local
2022/04/07 20:24:19 info done backup=2022-04-07 duration=24m11.34s operation=upload size=12.17GiB
Run clickhouse-backup delete local...
2022/04/07 20:24:20 info done backup=2022-04-07 duration=258ms location=local operation=delete
Done
Still waiting for data or an error? Try getting logs from the artifacts.
Logs do not appear for pods that are deleted.

Argo Events 應用

LHCI Farm

背景

什麼是 Lighthouse CI？

Lighthouse CI

Lighthouse CI is a suite of tools that make continuously running, saving, retrieving, and asserting agai nst Lighthouse results as easy as possible.

```yaml
name: CI
on: [push]
jobs:
  lighthouseci:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - uses: actions/setup-node@v1
        with:
          node-version: 14
      - run: npm install && npm install -g @lhci/cli@0.8.x
      - run: npm run build
      - run: lhci autorun
```


※Source from：https://github.com/GoogleChrome/lighthouse-ci

Example Builds

Filter find a build... 1 / 100

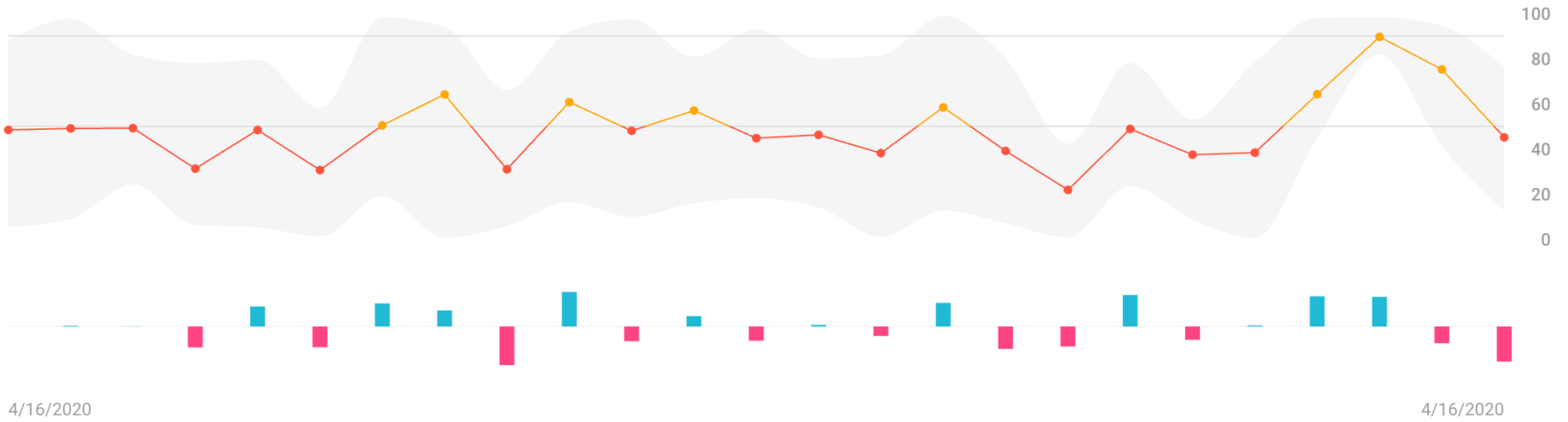
00000000	fix: 499	dev499	Mar 05 12:22 AM
00000000	feat: 498	master	Mar 04 10:09 PM
00000000	fix: 497	dev497	Mar 04 2:30 AM
00000000	feat: 496	master	Mar 03 6:42 AM
00000000	fix: 495	dev495	Mar 02 3:57 PM

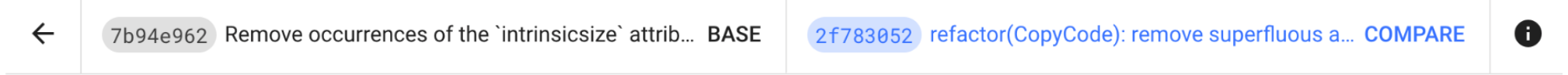
URL http://localhost:PORT/index.html Branch master

Performance

Timeline Distribution 25 50 100 MAX

Overview





URL http://localhost:PORT/javascript-and-google-search-io-2019/ Threshold 5%

Performance	Accessibility	Best Practices	SEO	PWA
55	90	79	96	PWA
+2	-8	+0	+0	-

This base build is not the exact ancestor of the compare. Differences may not be due to this specific commit.

△ 0-49 □ 50-89 ○ 90-100 ■ Regression ■ Improvement

Metrics

△ Max Potential First Input Delay	300	+56 ms	600
□ First Meaningful Paint	2.0	+209 ms	4.0
○ First Contentful Paint	1.0	-440 ms	4.0
○ Speed Index	1.0	-271 ms	4.0

Opportunities

□ Eliminate render-blocking resources 2 → 1

Argo Events

事件驅動的工作流自動化框架

Argo Events

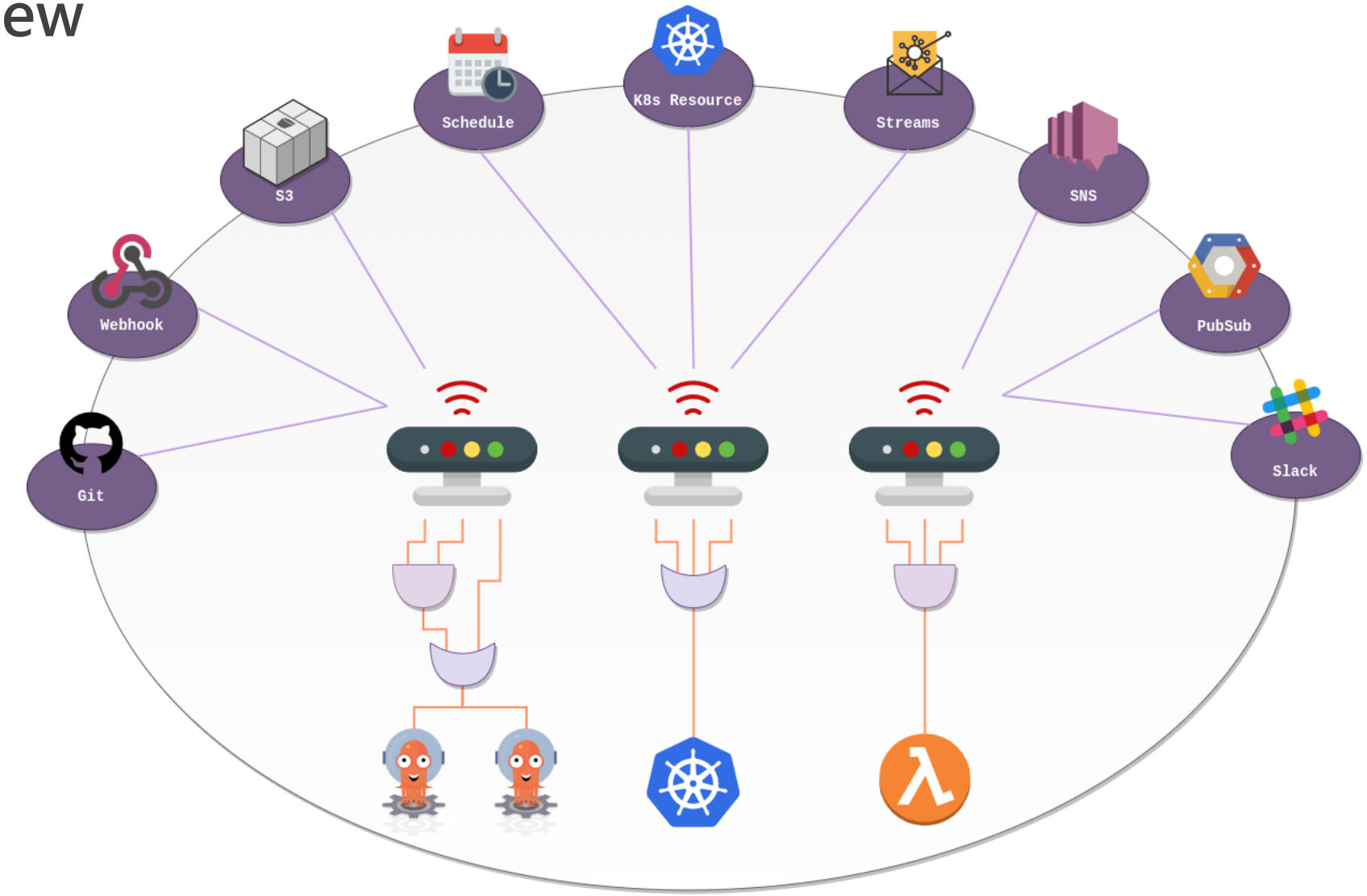
什麼是 Argo Events ?

Argo Events是一個用於 Kubernetes 的事件驅動的工作流自動化框架，可幫助您觸發來自各種來源 (如 webhook、S3、調度、消息隊列、gcp pubsub、sns、sqs) 的事件的 K8s 對象、Argo 工作流、無服務器工作負載等.....

- 支持來自 20 多個事件源的事件。
- 能夠為工作流自動化定制業務級約束邏輯。
- 管理從簡單、線性、實時到復雜、多源事件的所有內容。
- 支持 Kubernetes Objects、Argo Workflow、AWS Lambda、Serverless 等作為觸發器。

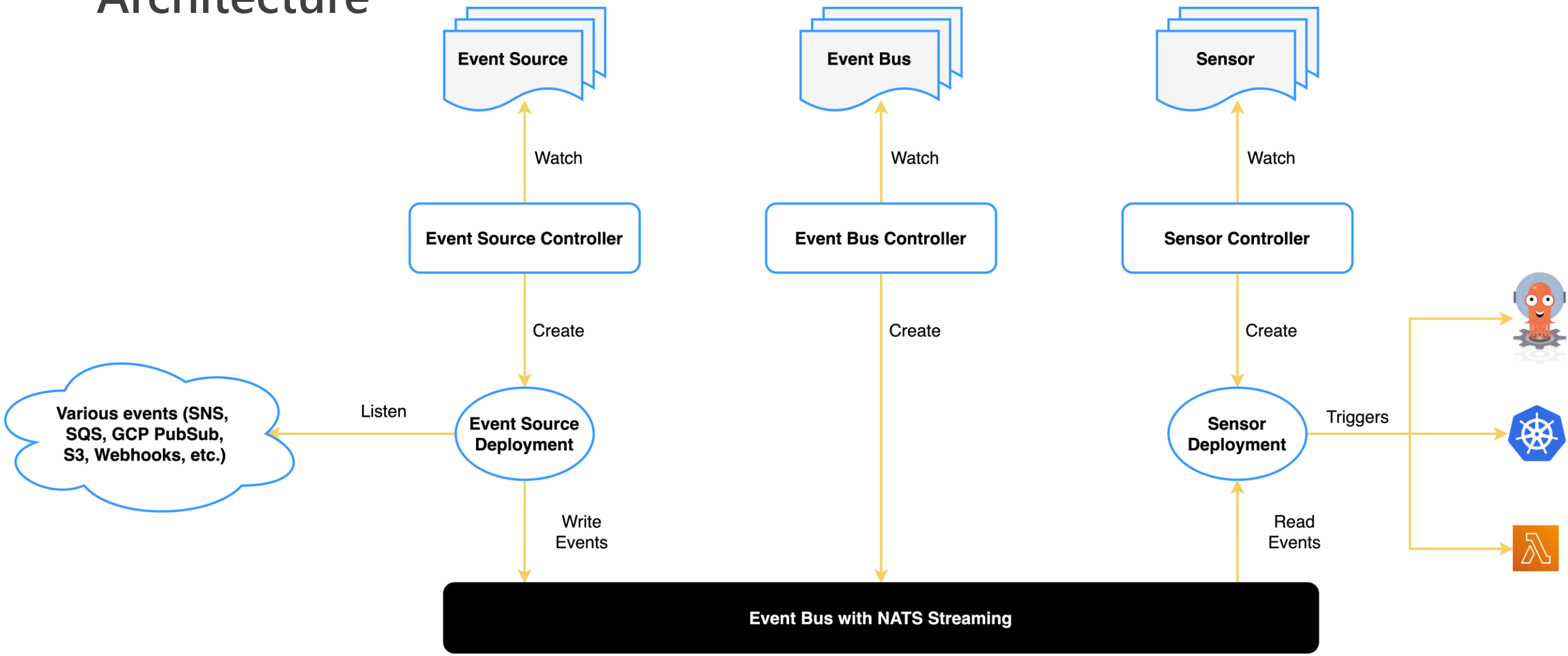
※Source from：https://argoproj.github.io/argo-events/

Overview



※Source from : https://argoproj.github.io/argo-events/

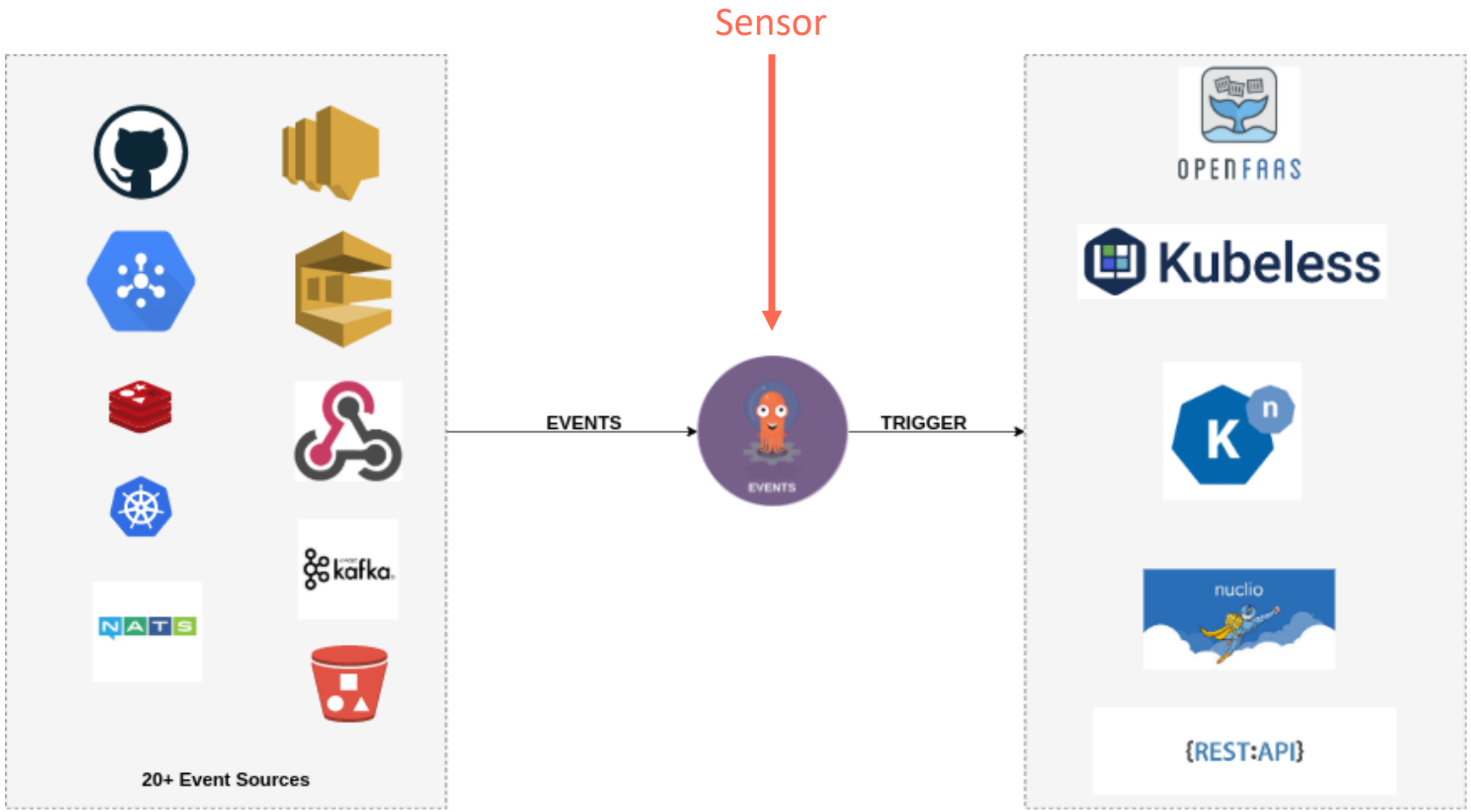
Architecture



※Source from : https://argoproj.github.io/argo-events/

EventSource & Sensors

HTTP Trigger



EventSource

Support types

AMQP
AWS SNS
AWS SQS
Azure Events Hub
Bitbucket
Bitbucket Server
Calendar
Emitter
File Based Events
GCP PubSub
Generic EventSource
GitHub
GitLab
HDFS
K8s Resources
Kafka
Minio
NATS
NetApp StorageGrid
MQTT
NSQ
Pulsar
Redis
Slack
Stripe
Webhooks

※Source from：https://argoproj.github.io/argo-events/concepts/event_source/

EventSource

EventSource With Multiple Events

```yaml
apiVersion: argoproj.io/v1alpha1
kind: EventSource
metadata:
  name: mixed-sources
spec:
  webhook:
    webhook-example: # eventName
      port: "12000"
      endpoint: /example
      method: POST
  sns:
    sns-example: # eventName
      topicArn: arn:aws:sns:us-east-1:XXXXXXXX:test
      webhook:
        endpoint: "/"
        port: "15000"
      accessKey:
        key: my-key
        name: my-name
      secretKey:
        key: my-secret-key
        name: my-secret-name
      region: us-east-1
```

```
kubectl -n lhci-farm get deploy
NAME                          READY   UP-TO-DATE   AVAILABLE   AGE
webhook-eventsource-975pr     2/2     2            2           134d
```

Sensors

Trigger

AWS Lambda
Apache OpenWhisk
Argo Rollouts
Argo Workflows
Custom - Build Your Own
HTTP Requests - Serverless Workloads (OpenFaas, Kubeless, Knative etc.)
Kafka Messages
NATS Messages
Slack Notifications
Azure Event Hubs Messages
Create any Kubernetes Objects
Log (for debugging event bus messages)

※Source from：https://argoproj.github.io/argo-events/concepts/trigger/

Sensors

HTTP Trigger

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Sensor
metadata:
  name: minio
spec:
  dependencies:
    - name: test-dep
      eventSourceName: minio
      eventName: example
  triggers:
    - template:
        name: http-trigger
        http:
          url: http://http-server.argo-events.svc:8090/hello
          payload:
            - src:
                dependencyName: test-dep
                dataKey: notification.0.s3.bucket.name
              dest: bucket
            - src:
                dependencyName: test-dep
                contextKey: type
              dest: type
          method: POST
      retryStrategy:
        steps: 3
        duration: 3s
```

```
kubectl -n lhci-farm get deploy

NAME                        READY   UP-TO-DATE   AVAILABLE   AGE
webhook-sensor-dcbhd        2/2     2            2           134d
```

對應EventSource名稱&Event名稱

能從dependency中取資料傳遞

Sensors

Trigger Conditions

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Sensor
metadata:
  name: example
spec:
  dependencies:
    - name: dep01
      eventSourceName: webhook-a
      eventName: example01
    - name: dep02
      eventSourceName: webhook-a
      eventName: example02
   - name: dep03
     eventSourceName: webhook-b
     eventName: example03
  triggers:
    - template:
        conditions: "dep02"
        name: trigger01
        http:
          url: http://abc.com/hello1
          method: GET
    - template:
        conditions: "dep02 && dep03"
        name: trigger02
        http:
          url: http://abc.com/hello2
          method: GET
    - template:
        conditions: "(dep01 || dep02) && dep03"
        name: trigger03
        http:
          url: http://abc.com/hello3
          method: GET
```

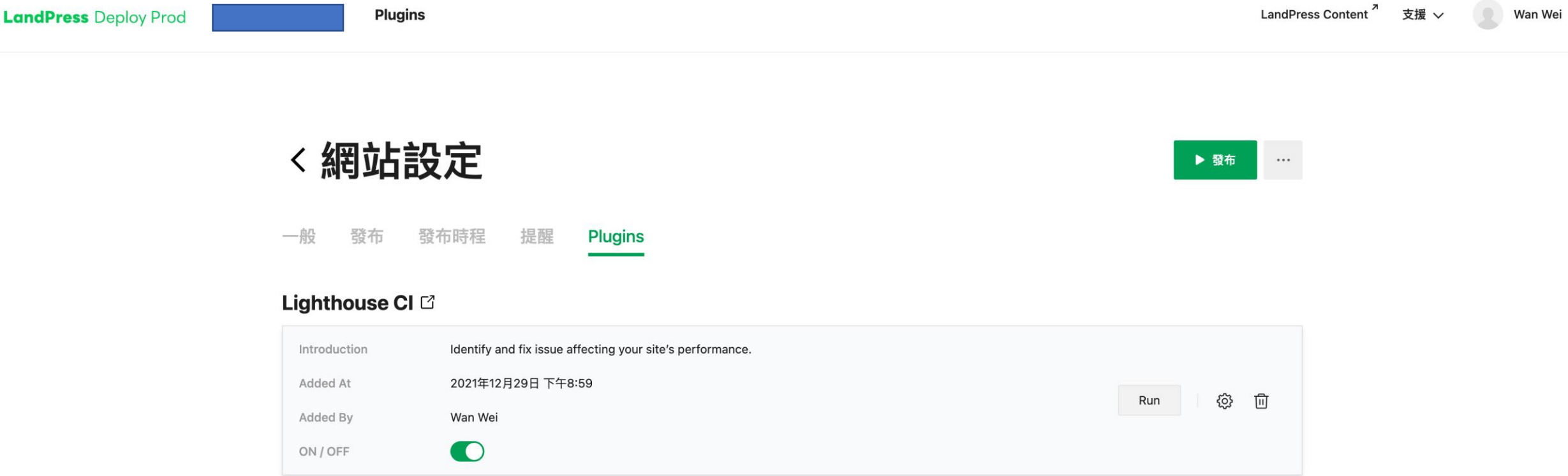
實戰

專案源起 - LINE LandPress Plugins

EventSource Webhook

Argo Workflow Trigger

Web UI

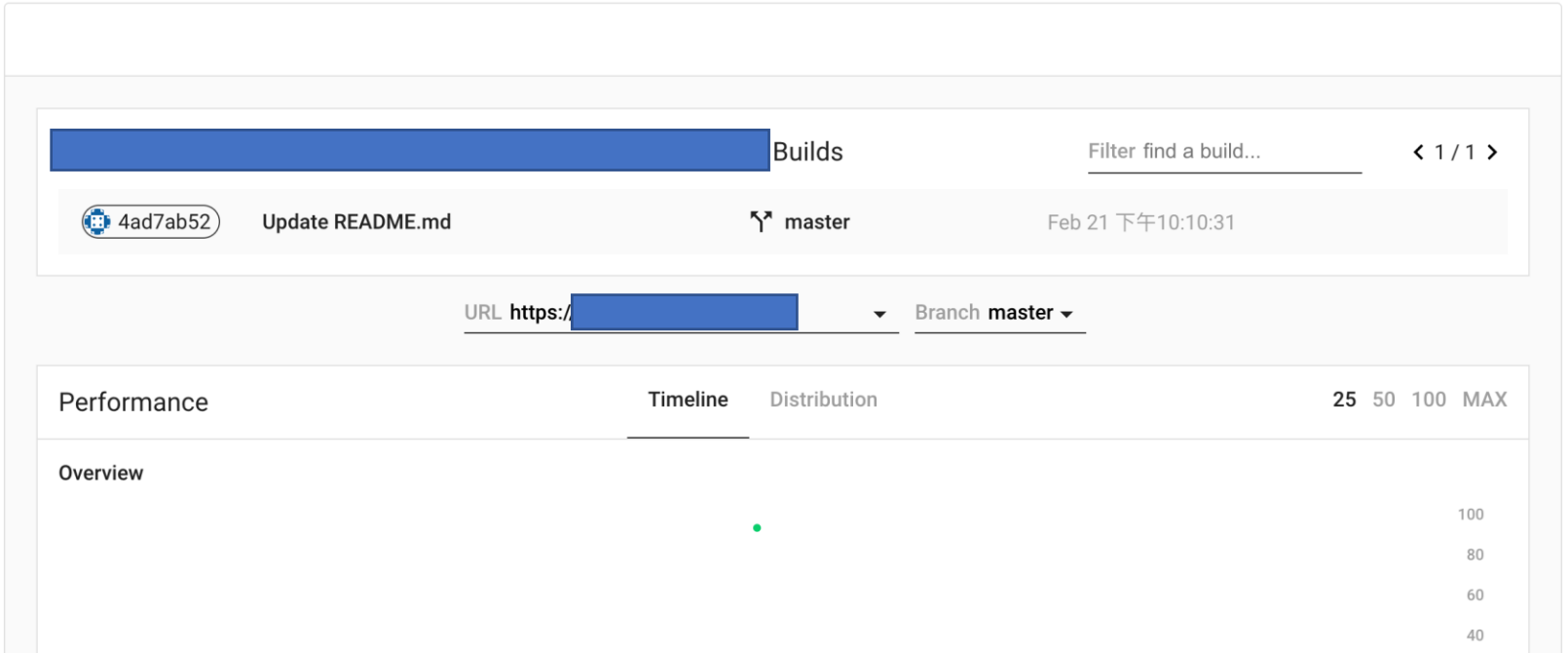

LandPress Deploy Prod
Plugins
LandPress Content
支援
Wan Wei
網站設定
發布
一般
發布
發布時程
提醒
Plugins
Lighthouse CI
Introduction
Identify and fix issue affecting your site's performance.
Added At
2021年12月29日 下午8:59
Added By
Wan Wei
ON / OFF
Run

網站設定

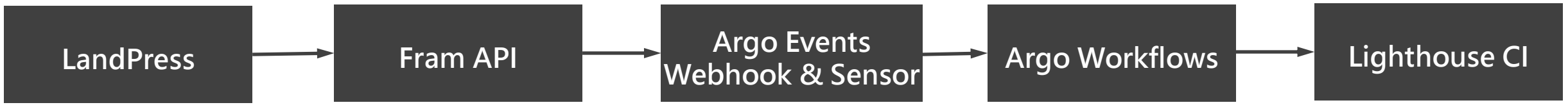
發布 ...

一般 發布 發布時程 提醒 Plugins

Lighthouse CI



架構



使用者點擊
觸發報表掃描

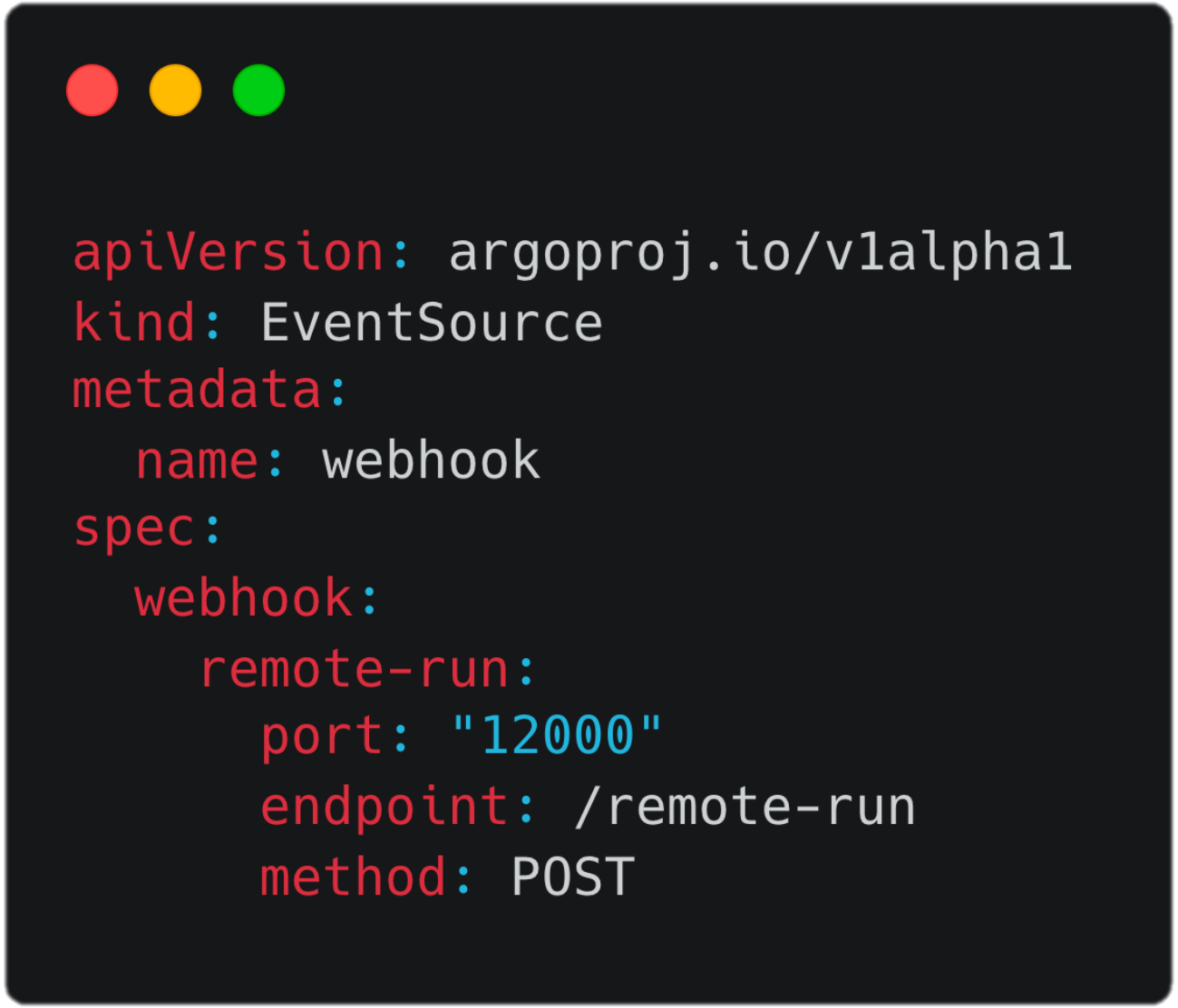
驗證資料
產生設定檔
紀錄Workflow狀態

收到Webhook
觸發Workflow

使用Farm API產生
的組態檔執行lhci
跑分

保存報告

EventSource



```yaml
apiVersion: argoproj.io/v1alpha1
kind: EventSource
metadata:
  name: webhook
spec:
  webhook:
    remote-run:
      port: "12000"
      endpoint: /remote-run
      method: POST
```

Sensor

觸發條件是Webhook Event Source

創建K8s資源

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Sensor
metadata:
  name: webhook
spec:
  dependencies:
  - name: webhook
    eventSourceName: webhook
    eventName: remote-run
  triggers:
  - template:
      name: webhook-workflow-trigger
      k8s:
        operation: create
        group: argoproj.io
        version: v1alpha1
        resource: workflows
        source:
          resource:
            apiVersion: argoproj.io/v1alpha1
```

呼叫Farm API回報狀態

執行lhci指令跑分

呼叫Farm API回報狀態

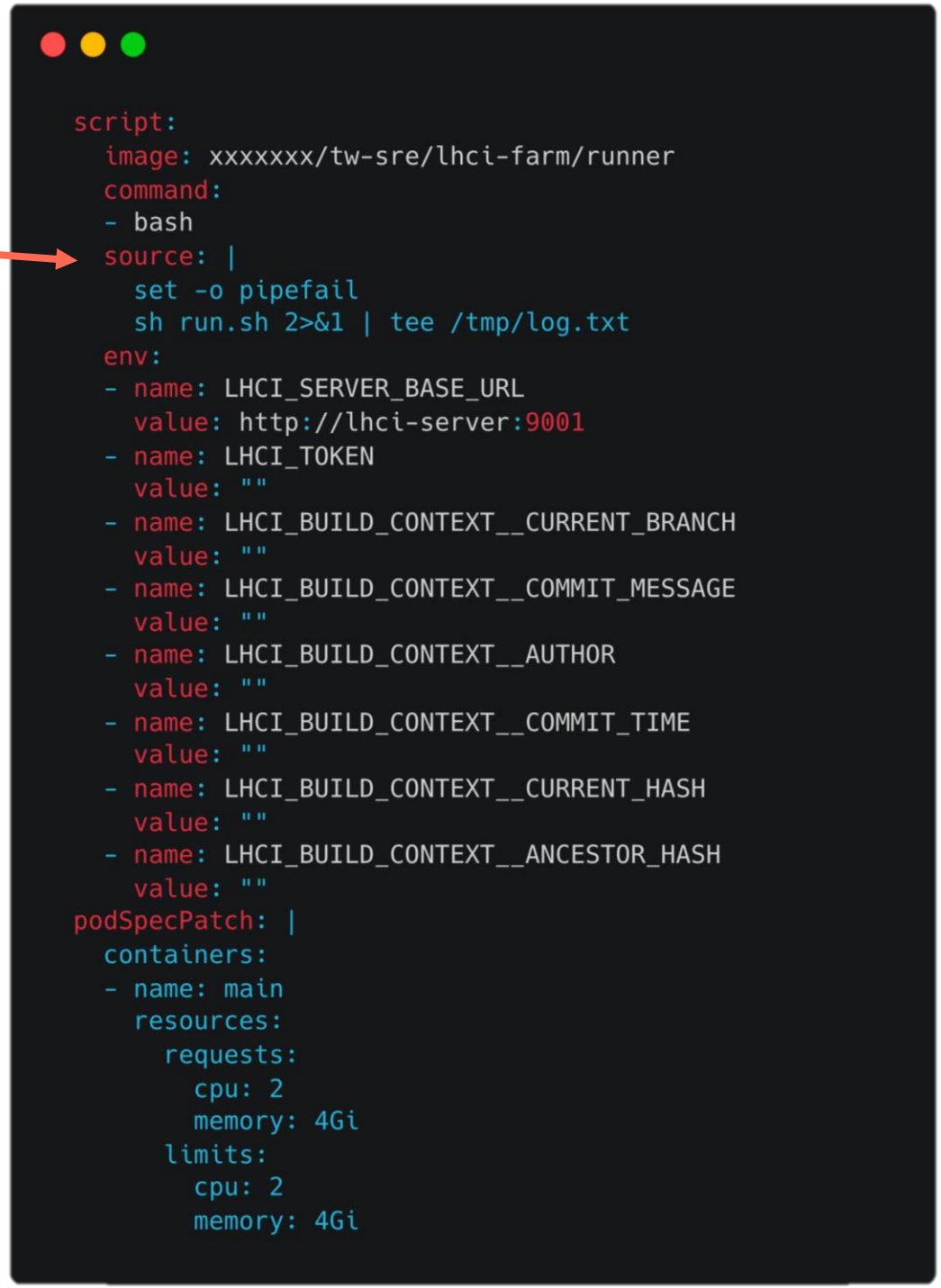
```yaml
templates:
- name: main
  steps:
  - - name: start
      template: callback
      arguments:
        parameters:
        - name: status
          value: Running
    - name: run
      template: run
      continueOn:
        error: true
        failed: true
  - - name: result
      template: callback
      arguments:
        parameters:
        - name: status
          value: "{{ steps.run.status }}"
```

```
- name: callback
  inputs:
    parameters:
    - name: runId
    - name: status
  http:
    url: http://farm-api:8080/v1/remote-runs/{{
inputs.parameters.runId }}/callback
    method: POST
    headers:
    - name: Content-Type
      value: application/json
    body: "{{ inputs.parameters }}"
```

從ObjectStorage上取得
跑分用組態檔

```yaml
- name: run
  inputs:
    parameters:
    - name: runId
    artifacts:
    - name: collects
      path: /home/lhci/reports/collects
      s3:
        endpoint: xxxxxxx
        insecure: false
        region: us-east-1
        bucket: twsre-cony-lhci-farm
        key: "{{ inputs.parameters.runId }}/collects"
        accessKeySecret:
          name: vos
          key: accesskey
        secretKeySecret:
          name: vos
          key: secretkey
    - name: upload
      path: /home/lhci/reports/upload.json
      s3:
        endpoint: xxxxxxx
        insecure: false
        region: us-east-1
        bucket: twsre-cony-lhci-farm
        key: "{{ inputs.parameters.runId }}/upload.json"
        accessKeySecret:
```

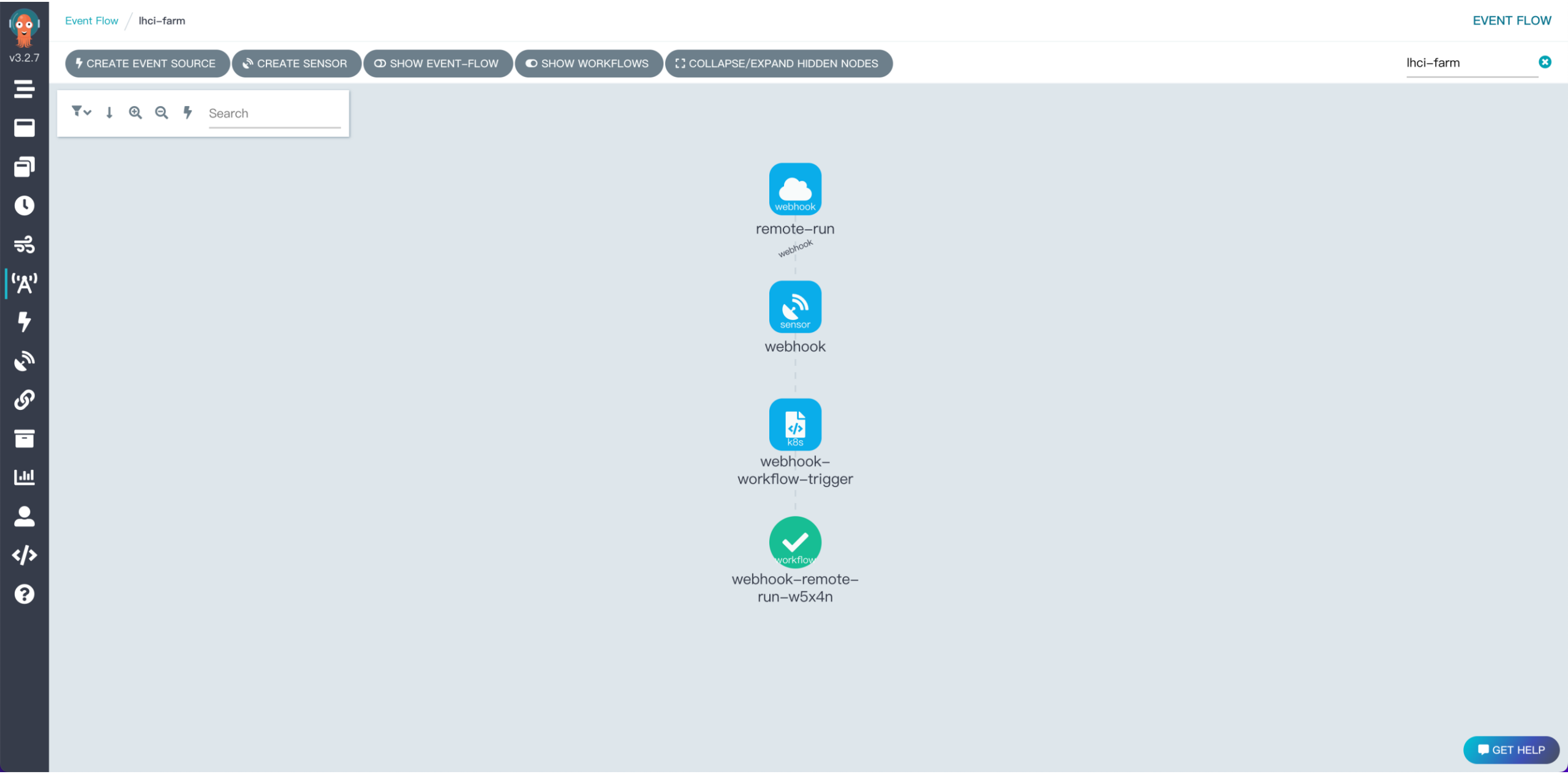
執行跑分

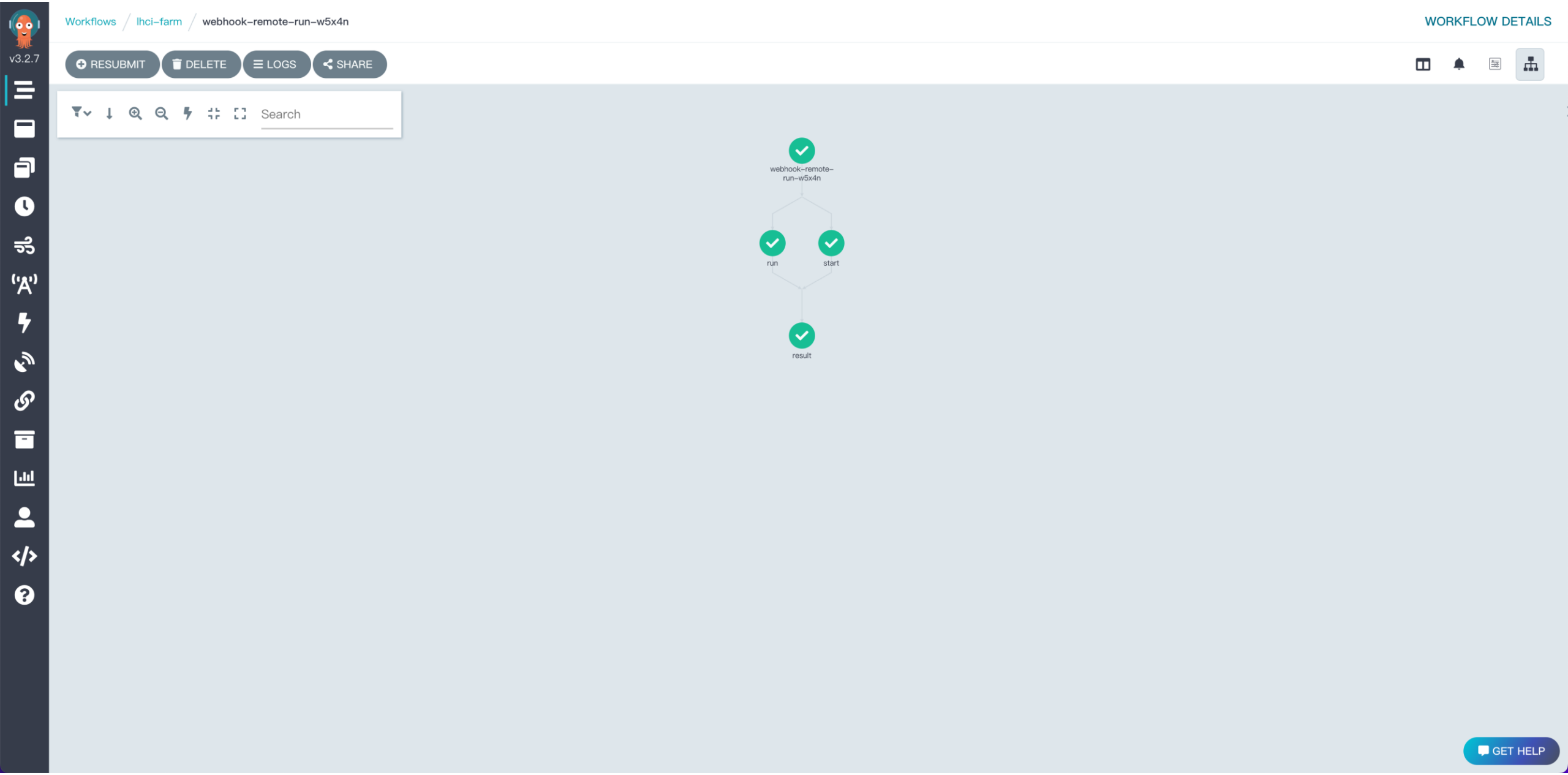


```
script:
  image: xxxxxxx/tw-sre/lhci-farm/runner
  command:
  - bash
  source: |
    set -o pipefail
    sh run.sh 2>&1 | tee /tmp/log.txt
  env:
  - name: LHCI_SERVER_BASE_URL
    value: http://lhci-server:9001
  - name: LHCI_TOKEN
    value: ""
  - name: LHCI_BUILD_CONTEXT__CURRENT_BRANCH
    value: ""
  - name: LHCI_BUILD_CONTEXT__COMMIT_MESSAGE
    value: ""
  - name: LHCI_BUILD_CONTEXT__AUTHOR
    value: ""
  - name: LHCI_BUILD_CONTEXT__COMMIT_TIME
    value: ""
  - name: LHCI_BUILD_CONTEXT__CURRENT_HASH
    value: ""
  - name: LHCI_BUILD_CONTEXT__ANCESTOR_HASH
    value: ""
podSpecPatch: |
  containers:
  - name: main
    resources:
      requests:
        cpu: 2
        memory: 4Gi
      limits:
        cpu: 2
        memory: 4Gi
```

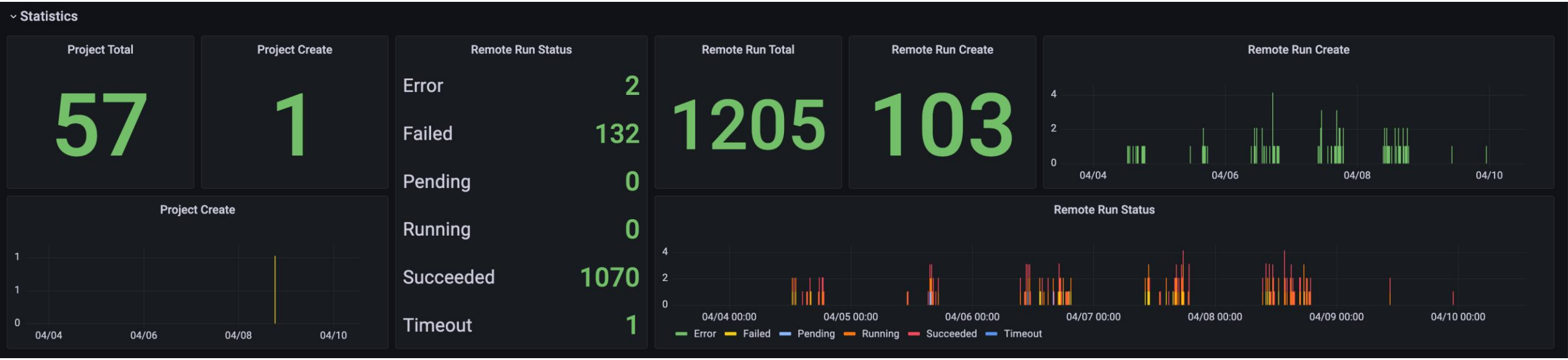
跑存跑分日誌

```yaml
outputs:
  artifacts:
  - name: log
    path: /tmp/log.txt
    archive:
      none: {}
    s3:
      endpoint: xxxxxx
      insecure: false
      region: us-east-1
      bucket: twsre-cony-lhci-farm
      key: "{{ inputs.parameters.runId }}/log.txt"
      accessKeySecret:
        name: vos
        key: accesskey
      secretKeySecret:
        name: vos
        key: secretkey
```


Event Flow
lhci-farm
EVENT FLOW
v3.2.7
CREATE EVENT SOURCE
CREATE SENSOR
SHOW EVENT-FLOW
SHOW WORKFLOWS
COLLAPSE/EXPAND HIDDEN NODES
lhci-farm
Search
webhook
remote-run
webhook
sensor
webhook
k8s
webhook-workflow-trigger
workflow
webhook-remote-run-w5x4n
GET HELP


v3.2.7
Workflows
lhci-farm
webhook-remote-run-w5x4n
WORKFLOW DETAILS
RESUBMIT
DELETE
LOGS
SHARE
Search
webhook-remote-run-w5x4n
run
start
result
GET HELP

使用情形


Statistics
Project Total
57
Project Create
1
Remote Run Status
Error 2
Failed 132
Pending 0
Running 0
Succeeded 1070
Timeout 1
Remote Run Total
1205
Remote Run Create
103
Remote Run Create
4
2
0
04/04
04/06
04/08
04/10
Project Create
1
1
0
04/04
04/06
04/08
04/10
Remote Run Status
4
2
0
04/04 00:00
04/05 00:00
04/06 00:00
04/07 00:00
04/08 00:00
04/09 00:00
04/10 00:00
Error
Failed
Pending
Running
Succeeded
Timeout

FAQ

TW SRE 招募中!!!