12th Symposium on Software Performance 2021

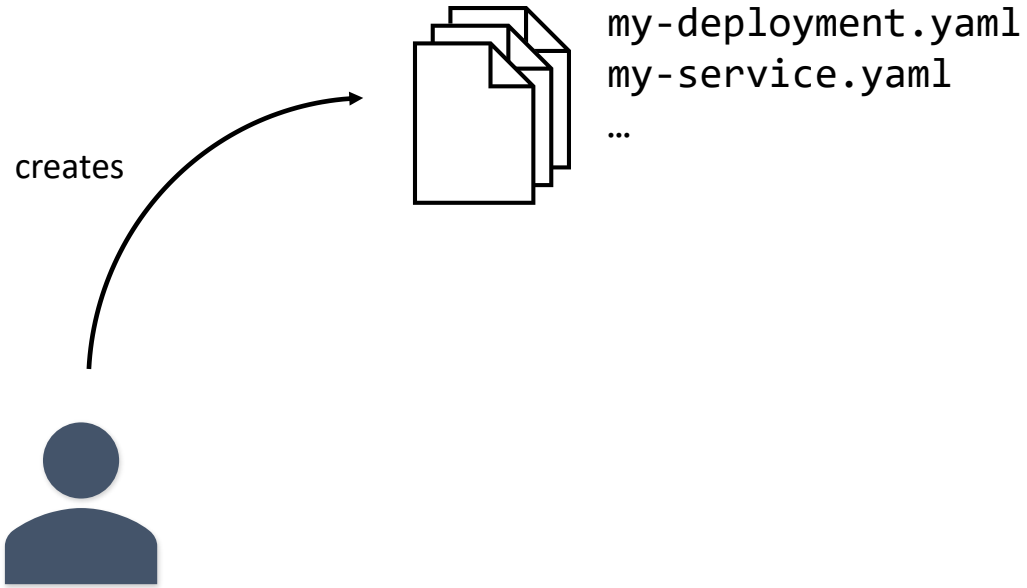
Reproducible Benchmarking of Cloud-Native Applications with the Kubernetes Operator Pattern

Sören Henning, Benedikt Wetzel and Wilhelm Hasselbring

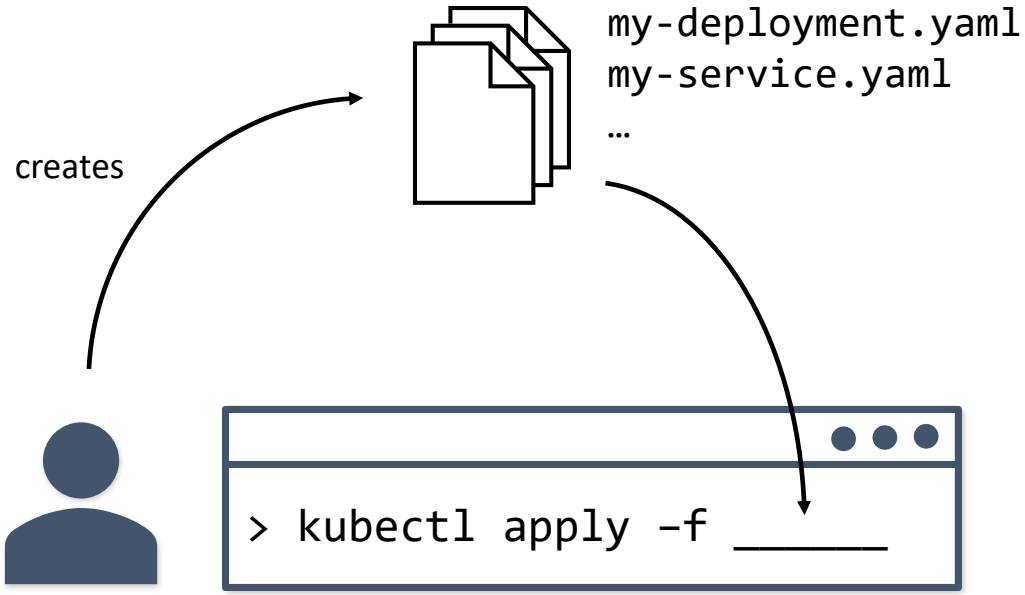
Kiel University
Christian-Albrechts-Universität zu Kiel

Cloud-native Applications in Kubernetes

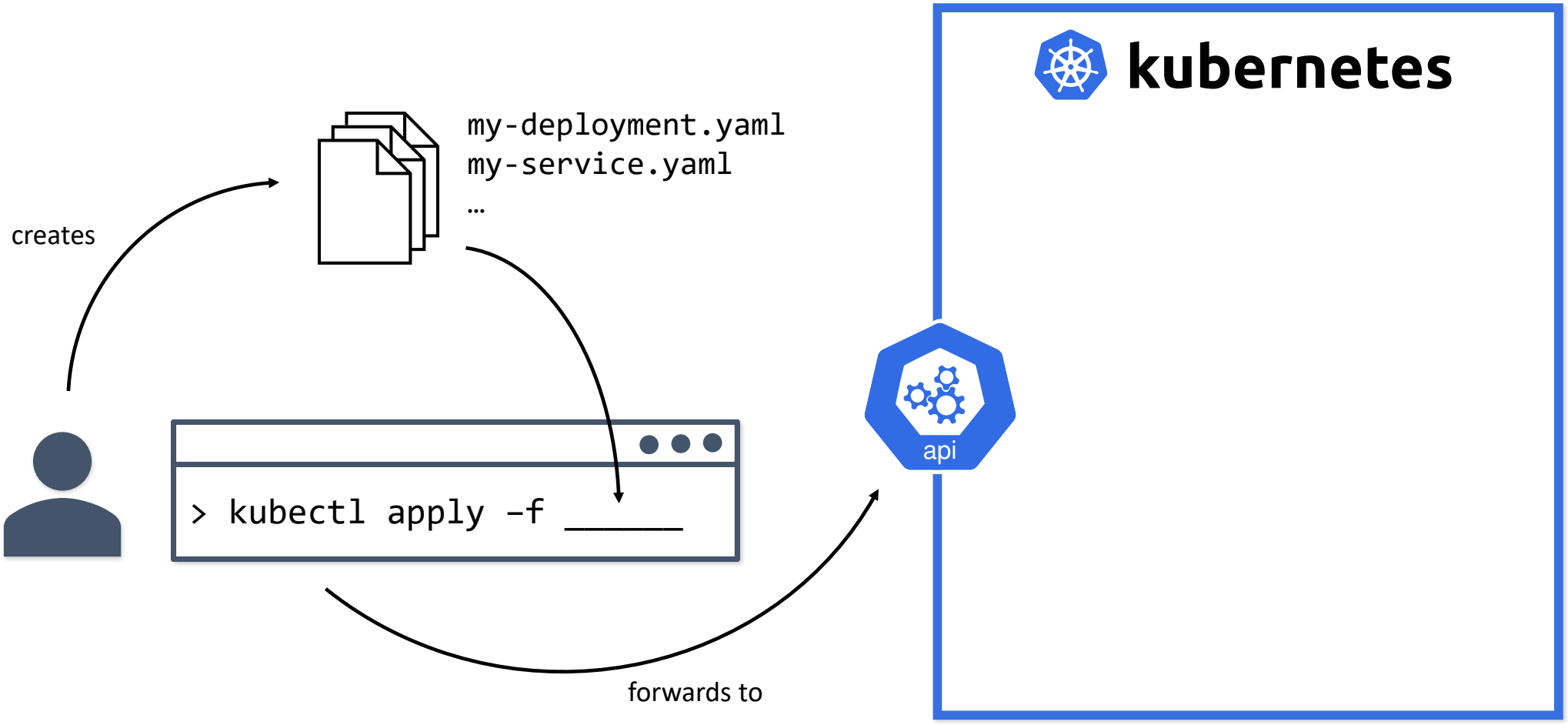
Cloud-native Applications in Kubernetes



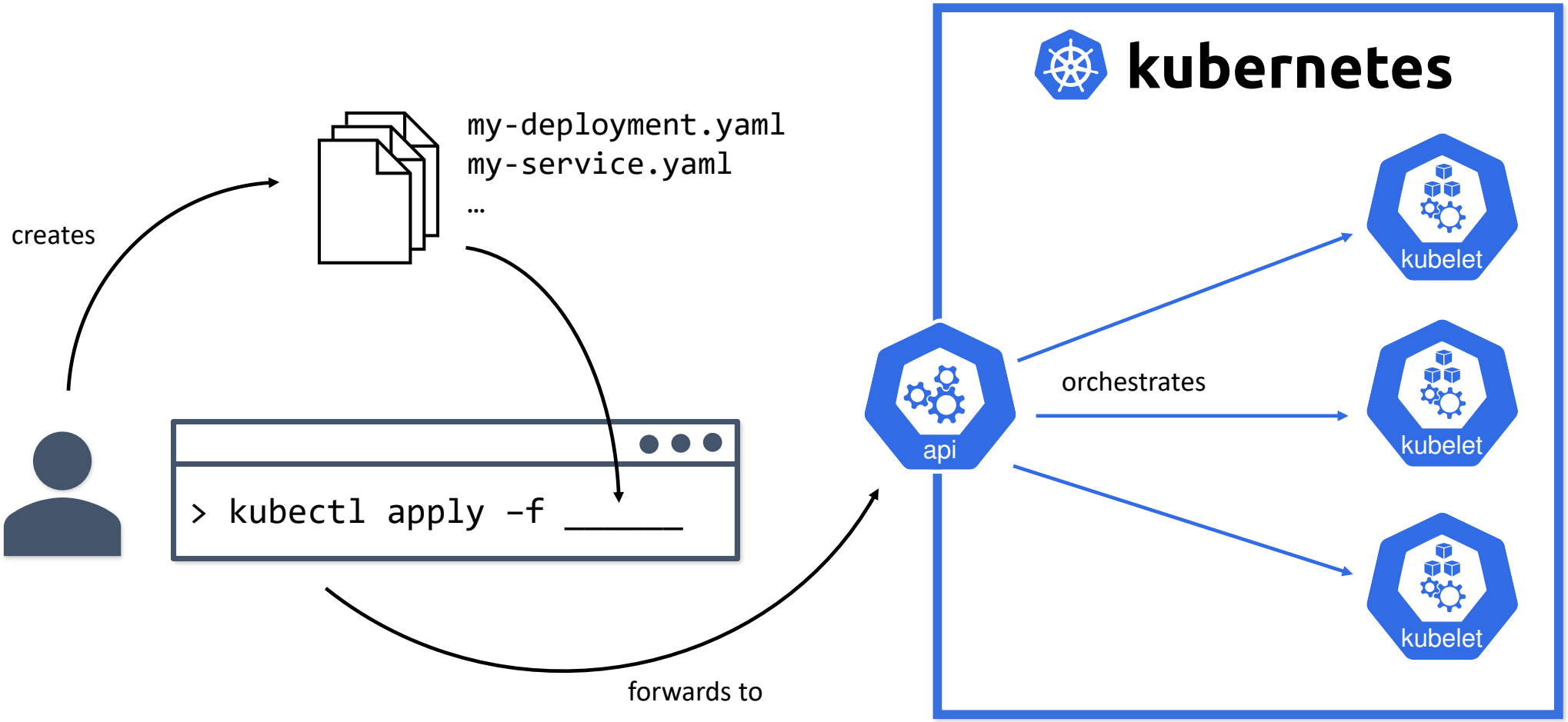
Cloud-native Applications in Kubernetes



Cloud-native Applications in Kubernetes



Cloud-native Applications in Kubernetes

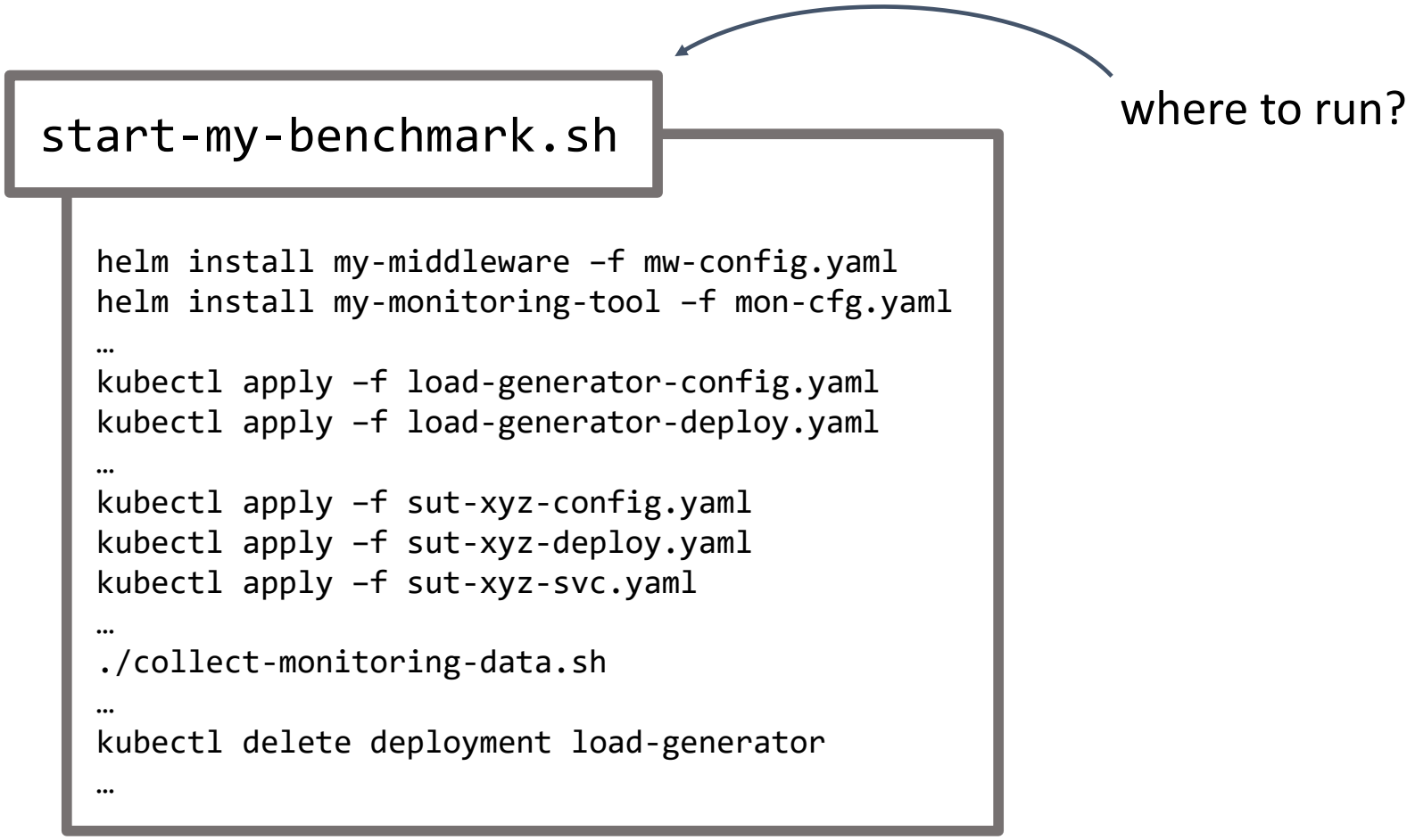


Benchmarking Cloud-native Applications: The Classical Way

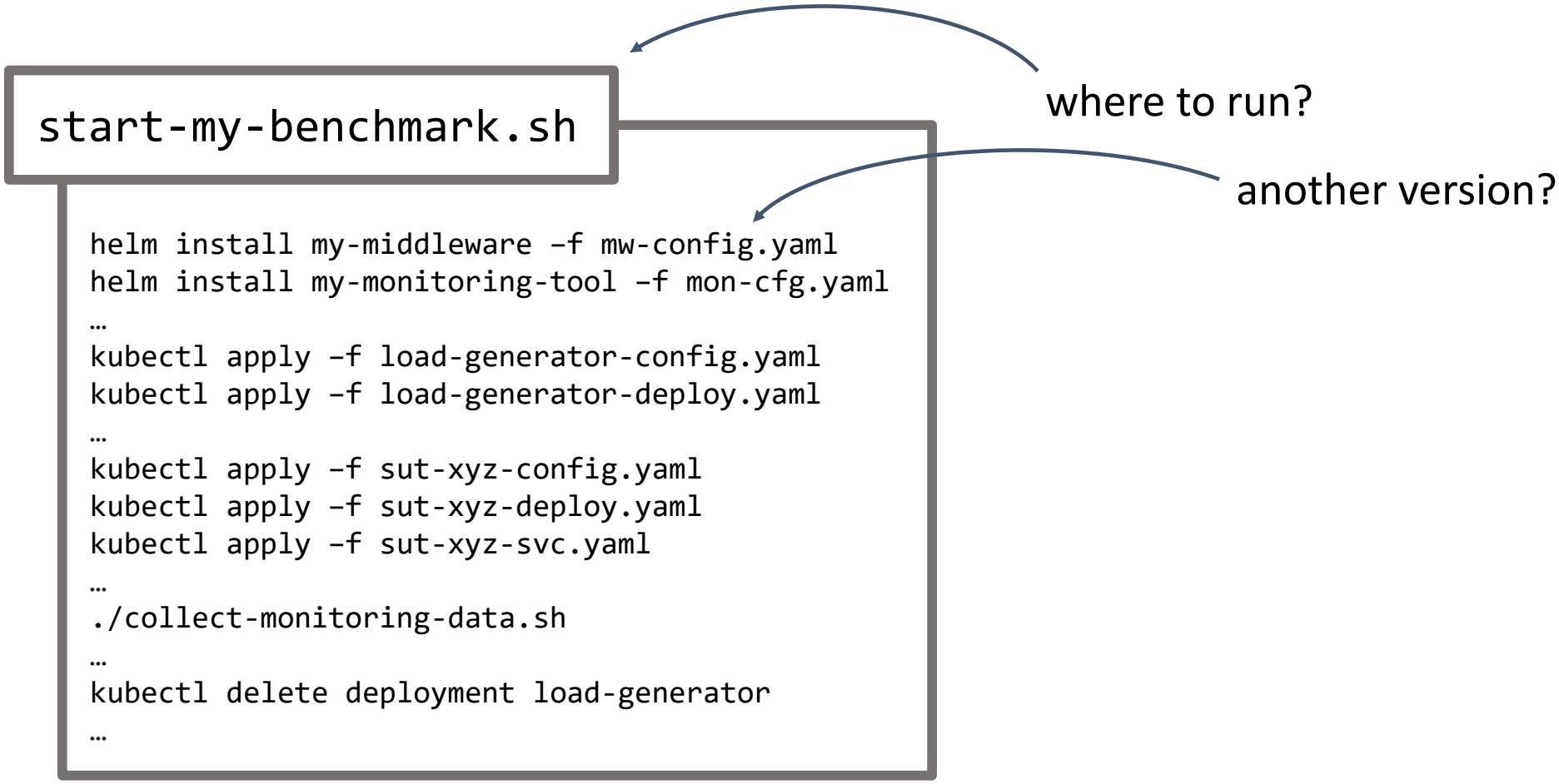
start-my-benchmark.sh

```
helm install my-middleware –f mw-config.yaml
helm install my-monitoring-tool –f mon-cfg.yaml
…
kubectl apply –f load-generator-config.yaml
kubectl apply –f load-generator-deploy.yaml
…
kubectl apply –f sut-xyz-config.yaml
kubectl apply –f sut-xyz-deploy.yaml
kubectl apply –f sut-xyz-svc.yaml
…
./collect-monitoring-data.sh
…
kubectl delete deployment load-generator
…
```

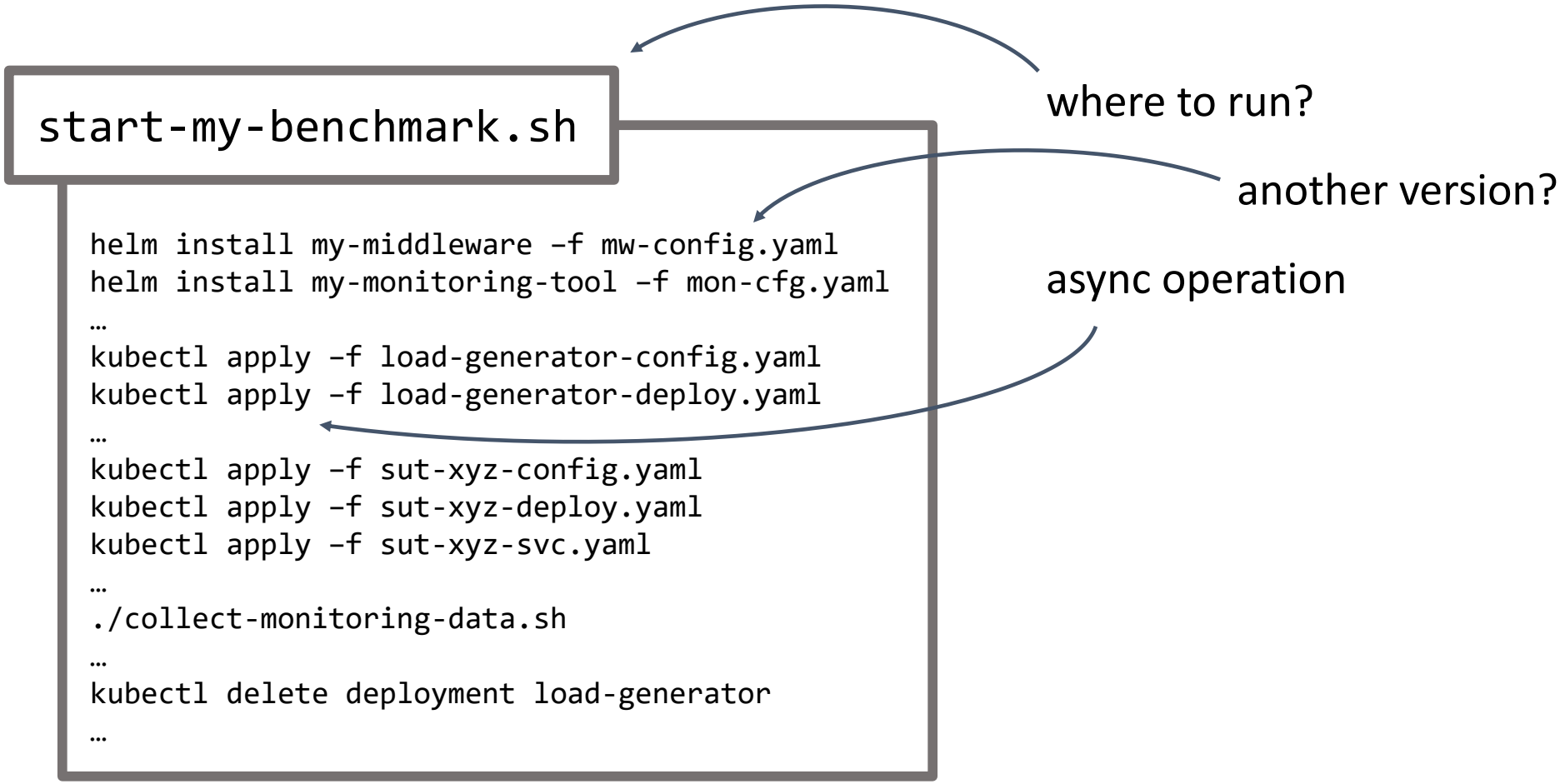
Benchmarking Cloud-native Applications: The Classical Way



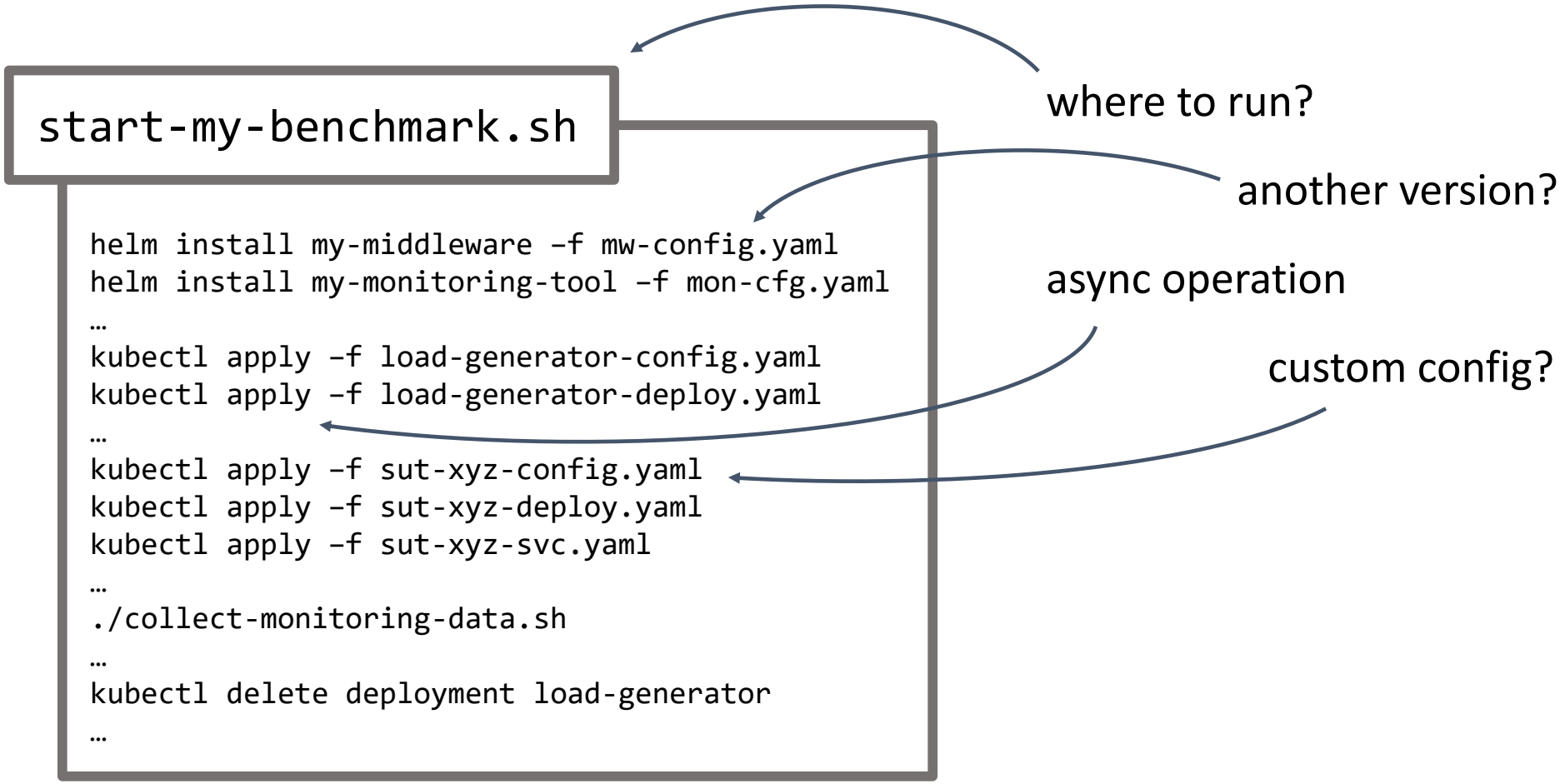
Benchmarking Cloud-native Applications: The Classical Way



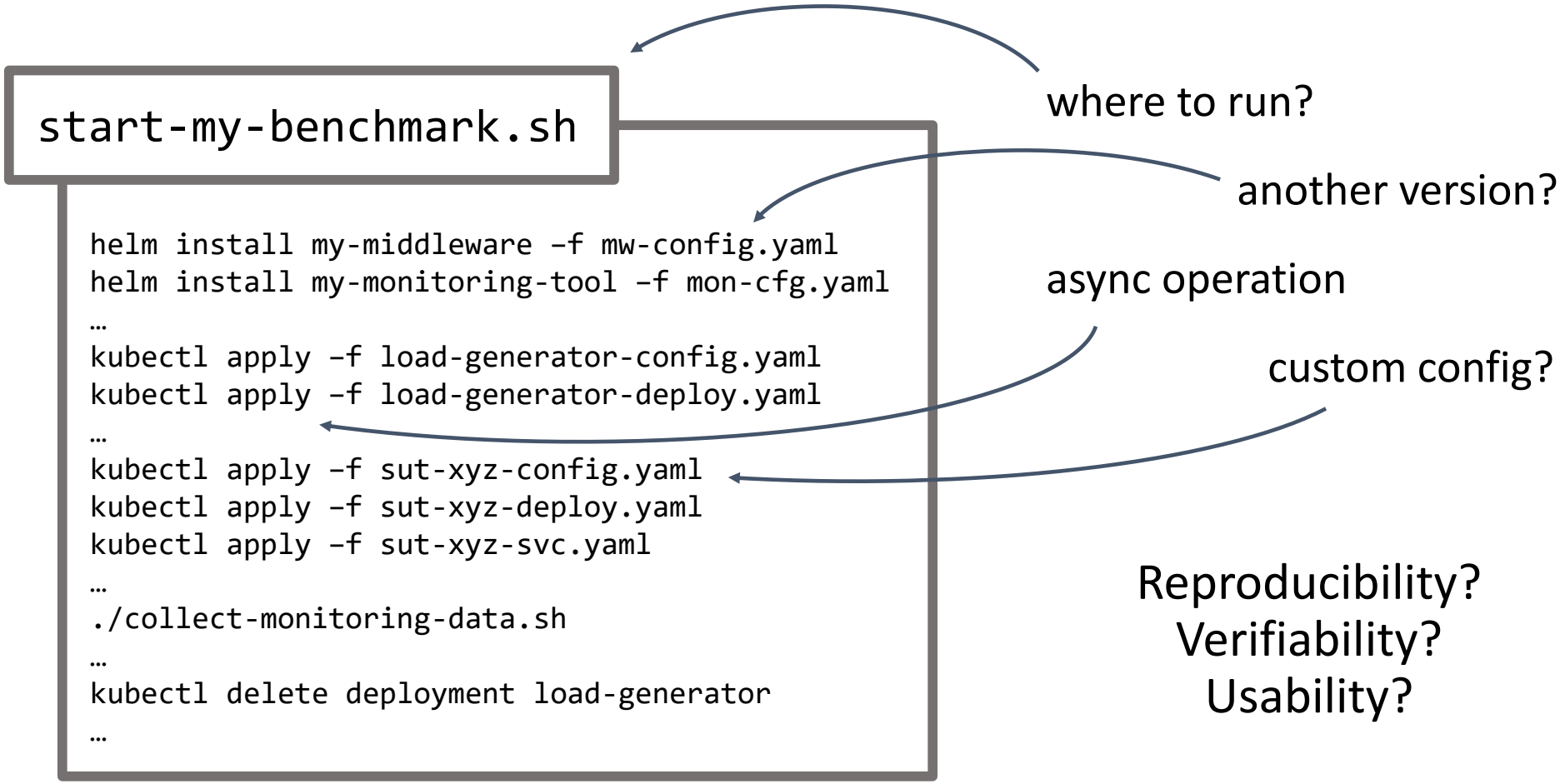
Benchmarking Cloud-native Applications: The Classical Way



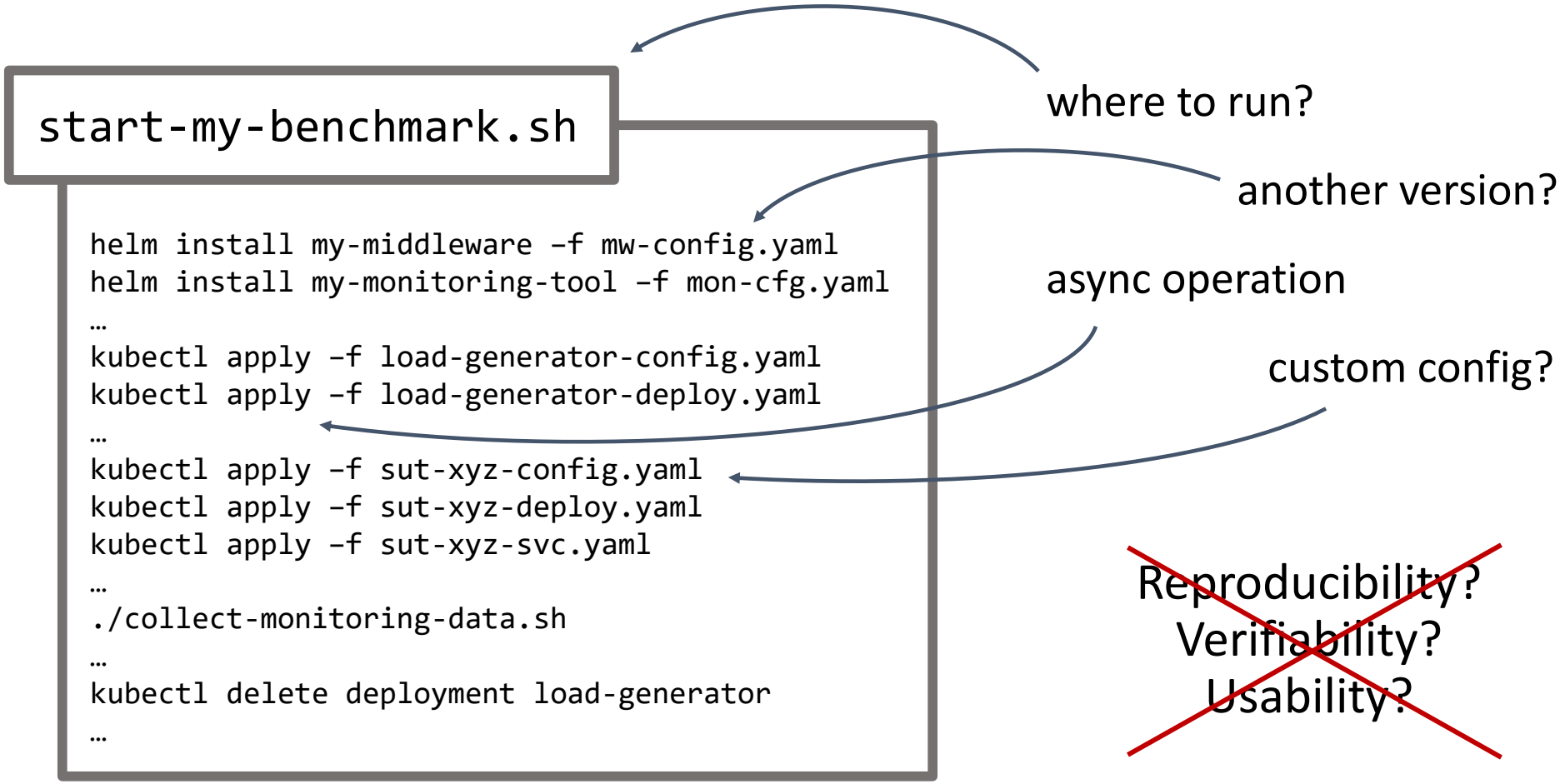
Benchmarking Cloud-native Applications: The Classical Way



Benchmarking Cloud-native Applications: The Classical Way



Benchmarking Cloud-native Applications: The Classical Way



Benchmarking Cloud-native Applications: The Classical Way

`start-my-benchmark.sh`

```
helm install my-middleware –f mw-config.yaml
helm install my-monitoring-tool –f mon-cfg.yaml
…
kubectl apply –f load-generator-config.yaml
kubectl apply –f load-generator-deploy.yaml
…
kubectl apply –f sut-xyz-config.yaml
kubectl apply –f sut-xyz-deploy.yaml
kubectl apply –f sut-xyz-svc.yaml
…
./collect-monitoring-data.sh
…
kubectl delete deployment load-generator
…
```

Suggestion:

Use established cloud-native tools

Benchmarking Cloud-native Applications: The Classical Way

`start-my-benchmark.sh`

```
helm install my-middleware –f mw-config.yaml
helm install my-monitoring-tool –f mon-cfg.yaml
…
kubectl apply –f load-generator-config.yaml
kubectl apply –f load-generator-deploy.yaml
…
kubectl apply –f sut-xyz-config.yaml
kubectl apply –f sut-xyz-deploy.yaml
kubectl apply –f sut-xyz-svc.yaml
…
./collect-monitoring-data.sh
…
kubectl delete deployment load-generator
…
```

Suggestion:

Use established cloud-native tools

CLOUD NATIVE
COMPUTING FOUNDATION

Benchmarking Cloud-native Applications: The Classical Way

start-my-benchmark.sh

```
helm install my-middleware –f mw-config.yaml
helm install my-monitoring-tool –f mon-cfg.yaml
…
kubectl apply –f load-generator-config.yaml
kubectl apply –f load-generator-deploy.yaml
…
kubectl apply –f sut-xyz-config.yaml
kubectl apply –f sut-xyz-deploy.yaml
kubectl apply –f sut-xyz-svc.yaml
…
./collect-monitoring-data.sh
…
kubectl delete deployment load-generator
…
```

Suggestion:

Use established cloud-native tools

CLOUD NATIVE COMPUTING FOUNDATION

kubernetes

Prometheus

…

Benchmarking Cloud-native Applications: The Classical Way

start-my-benchmark.sh

```
helm install my-middleware –f mw-config.yaml
helm install my-monitoring-tool –f mon-cfg.yaml
…
kubectl apply –f load-generator-config.yaml
kubectl apply –f load-generator-deploy.yaml
…
kubectl apply –f sut-xyz-config.yaml
kubectl apply –f sut-xyz-deploy.yaml
kubectl apply –f sut-xyz-svc.yaml
…
./collect-monitoring-data.sh
…
kubectl delete deployment load-generator
…
```

Suggestion:

Use established cloud-native tools

CLOUD NATIVE COMPUTING FOUNDATION

kubernetes

Prometheus

HELM

…

Operator Pattern

Benchmarking Cloud-native Applications: The Classical Way

start-my-benchmark.sh

```
helm install my-middleware –f mw-config.yaml
helm install my-monitoring-tool –f mon-cfg.yaml
…
kubectl apply –f load-generator-config.yaml
kubectl apply –f load-generator-deploy.yaml
…
kubectl apply –f sut-xyz-config.yaml
kubectl apply –f sut-xyz-deploy.yaml
kubectl apply –f sut-xyz-svc.yaml
…
./collect-monitoring-data.sh
…
kubectl delete deployment load-generator
…
```

Suggestion:

Use established cloud-native tools

CLOUD NATIVE COMPUTING FOUNDATION

kubernetes

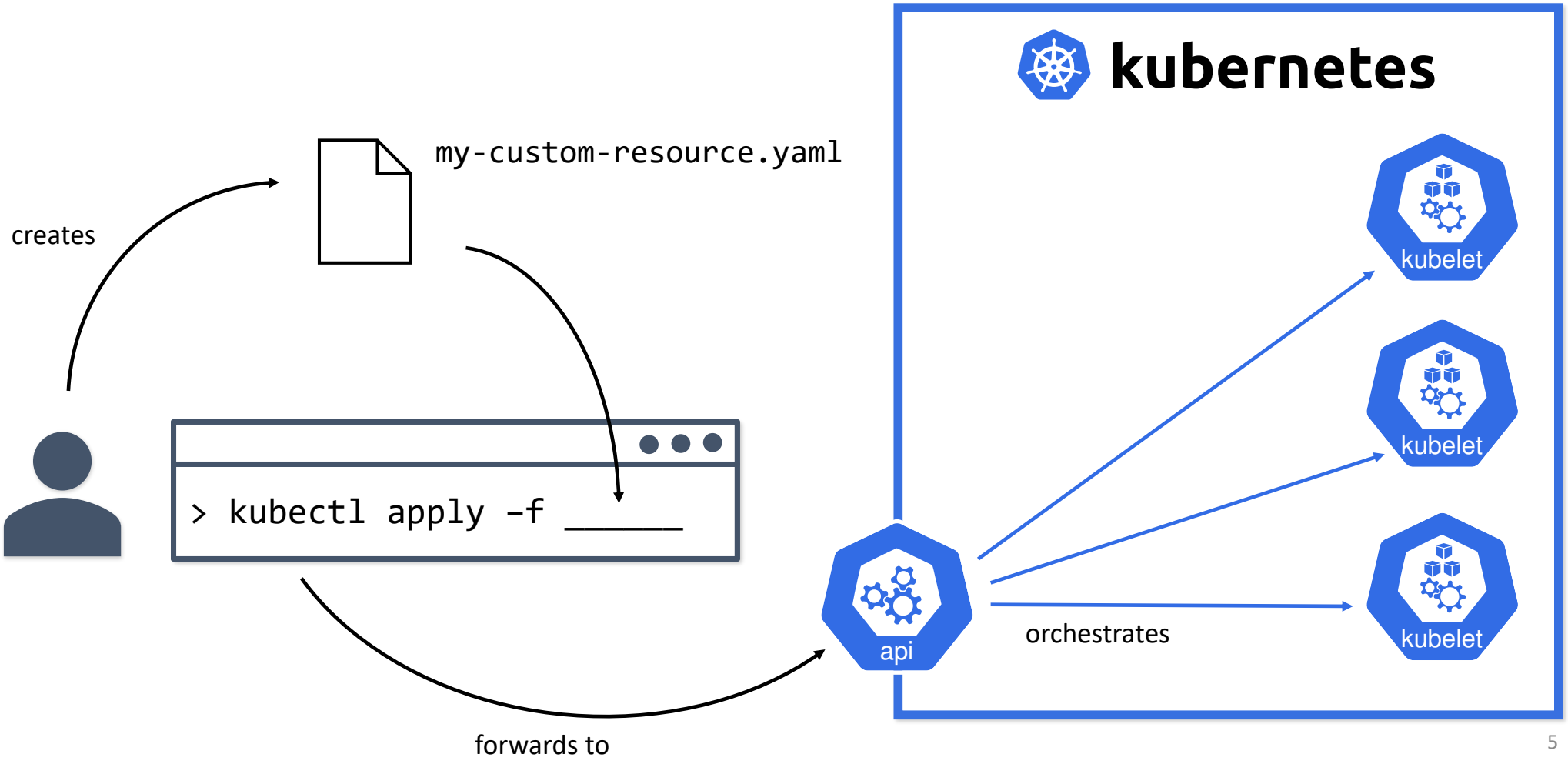
Prometheus

HELM

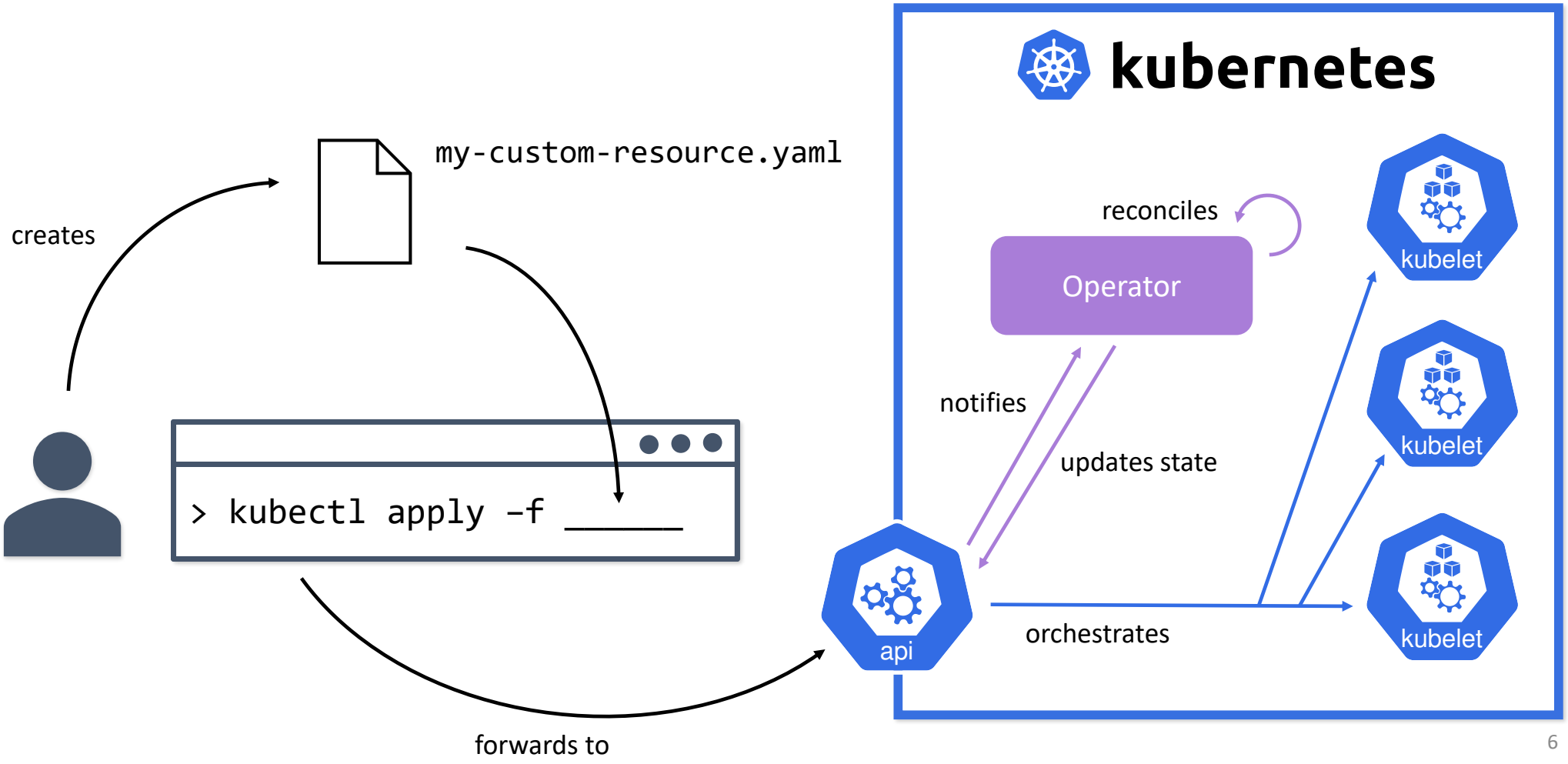
…

Operator Pattern

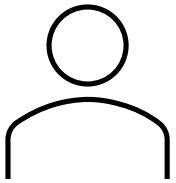
The Kubernetes Operator Pattern



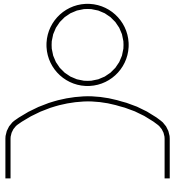
The Kubernetes Operator Pattern



Custom Resource Definitions (CRDs) for Benchmarking

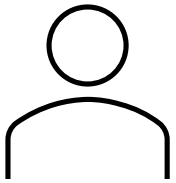


Benchmark Designer



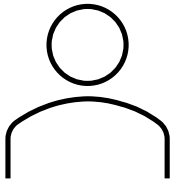
Benchmarker

Custom Resource Definitions (CRDs) for Benchmarking



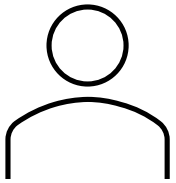
Benchmark Designer

- Expert regarding specific application types or software services
  e.g., researchers, engineers, standardization committees
- Defines how metrics and results can be interpreted to compare different SUTs
- Bundles all of this in a benchmarking tool or for a benchmarking tool



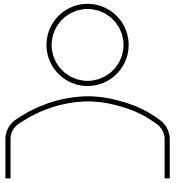
Benchmarker

Custom Resource Definitions (CRDs) for Benchmarking



Benchmark Designer

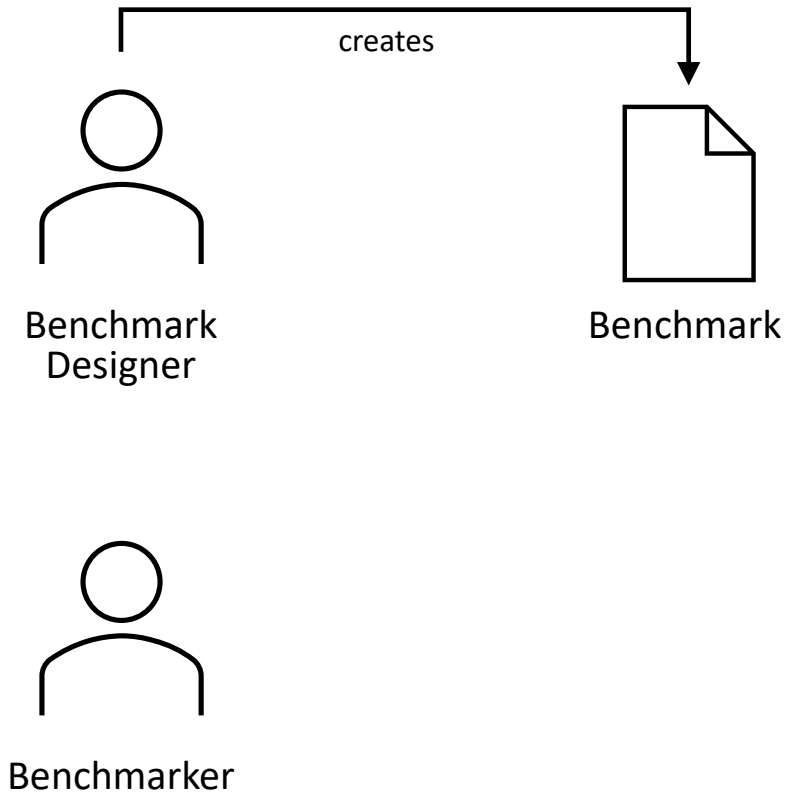
- Expert regarding specific application types or software services e.g., researchers, engineers, standardization committees
- Defines how metrics and results can be interpreted to compare different SUTs
- Bundles all of this in a benchmarking tool or for a benchmarking tool



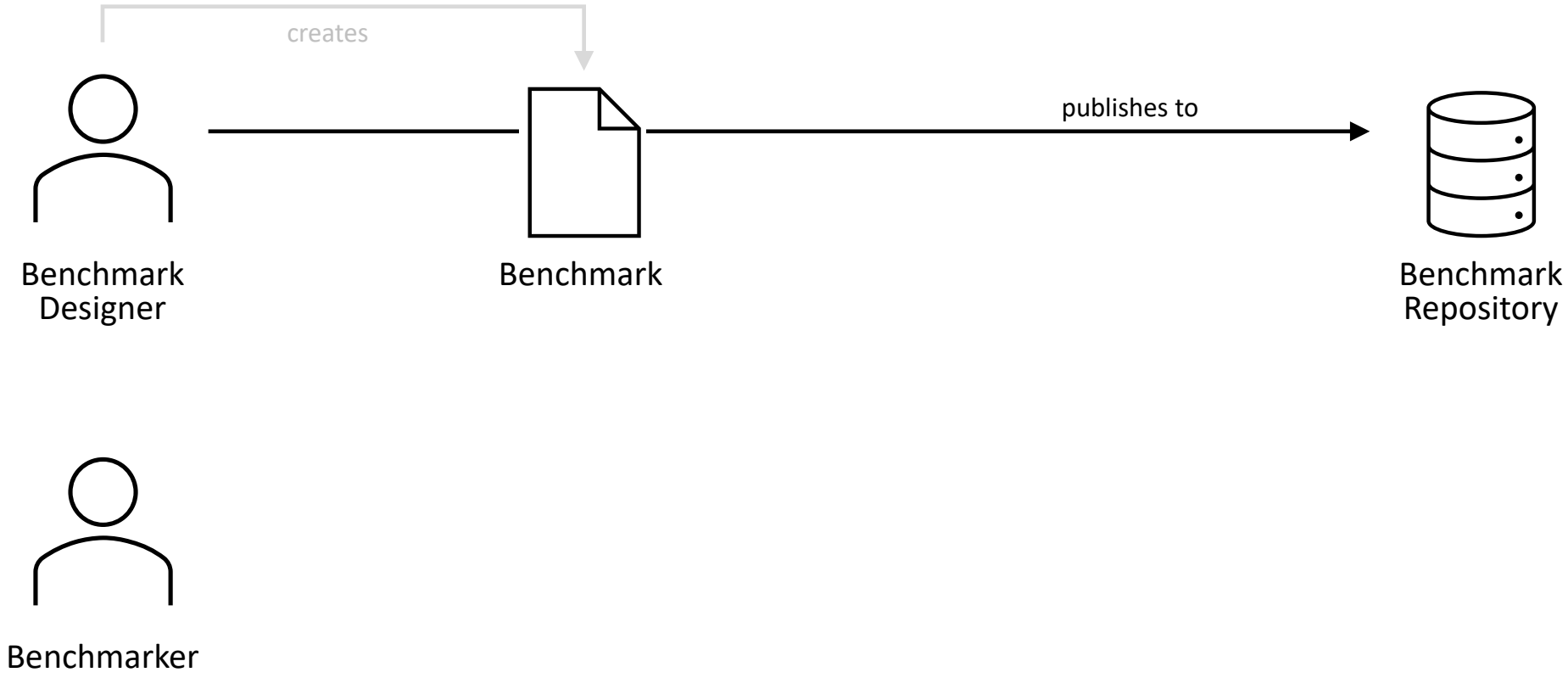
Benchmarker

- Compares and ranks different existing SUTs
- Evaluates new methods tools against a defined standard
- Repeats previous experiments
- Executes existing benchmarks

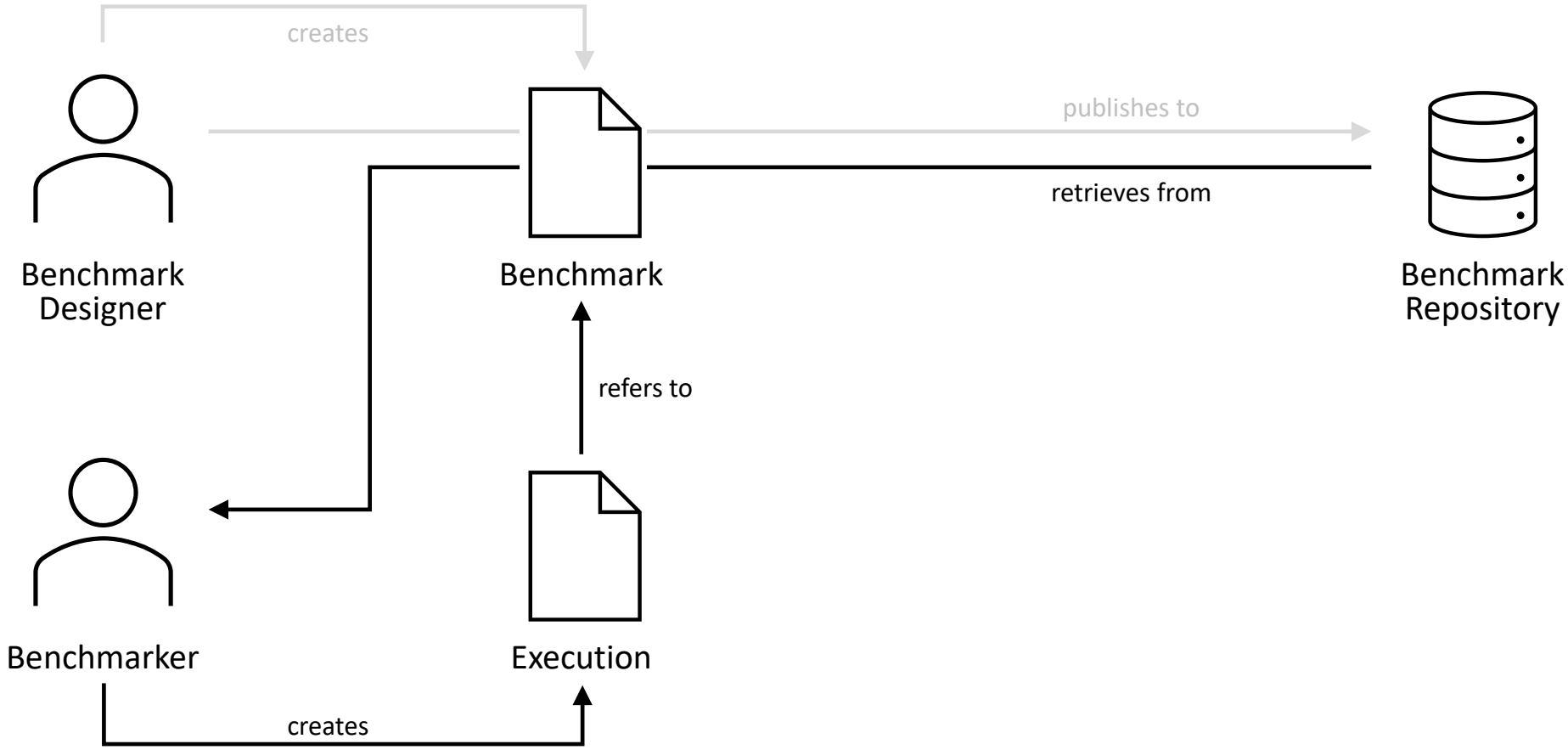
Custom Resource Definitions (CRDs) for Benchmarking



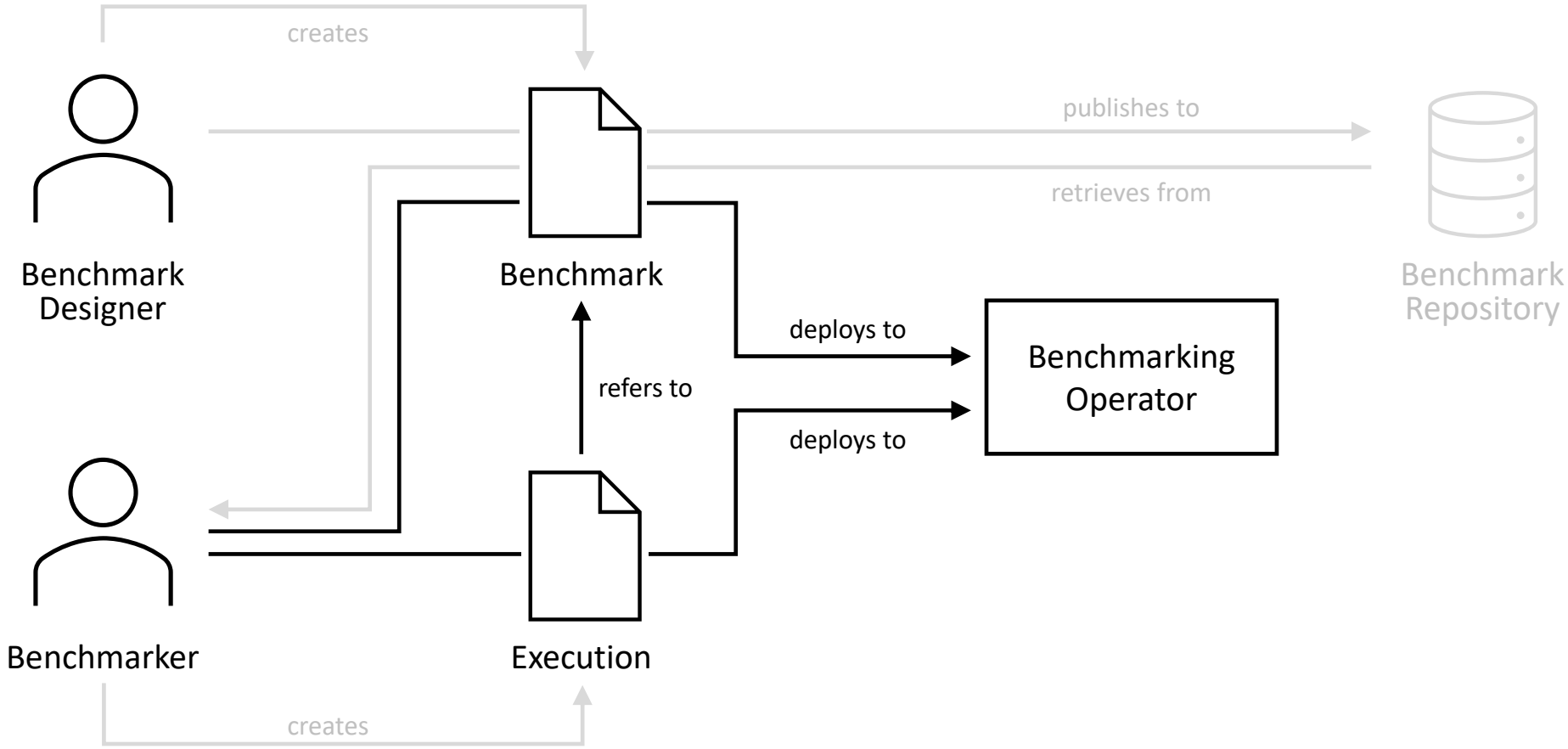
Custom Resource Definitions (CRDs) for Benchmarking



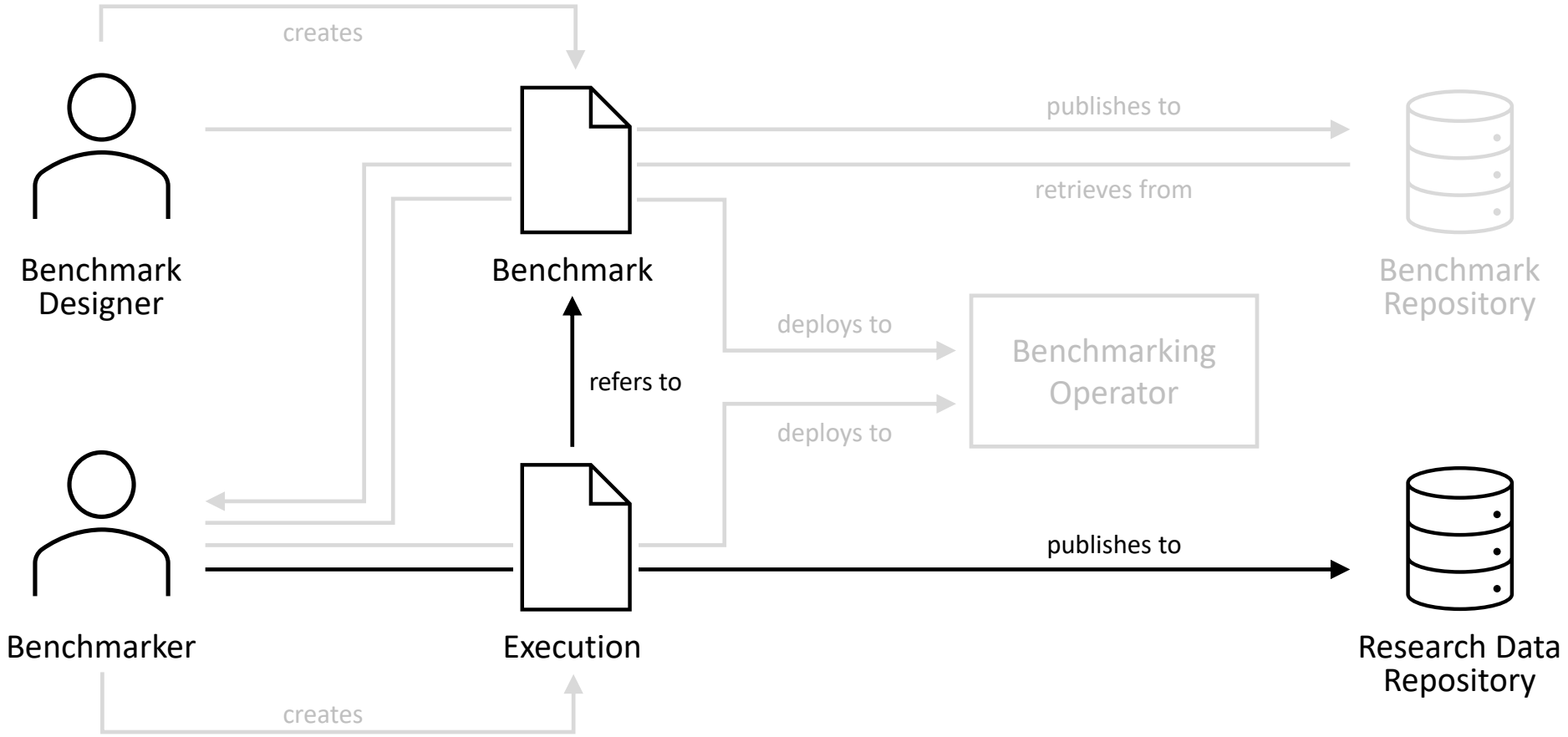
Custom Resource Definitions (CRDs) for Benchmarking



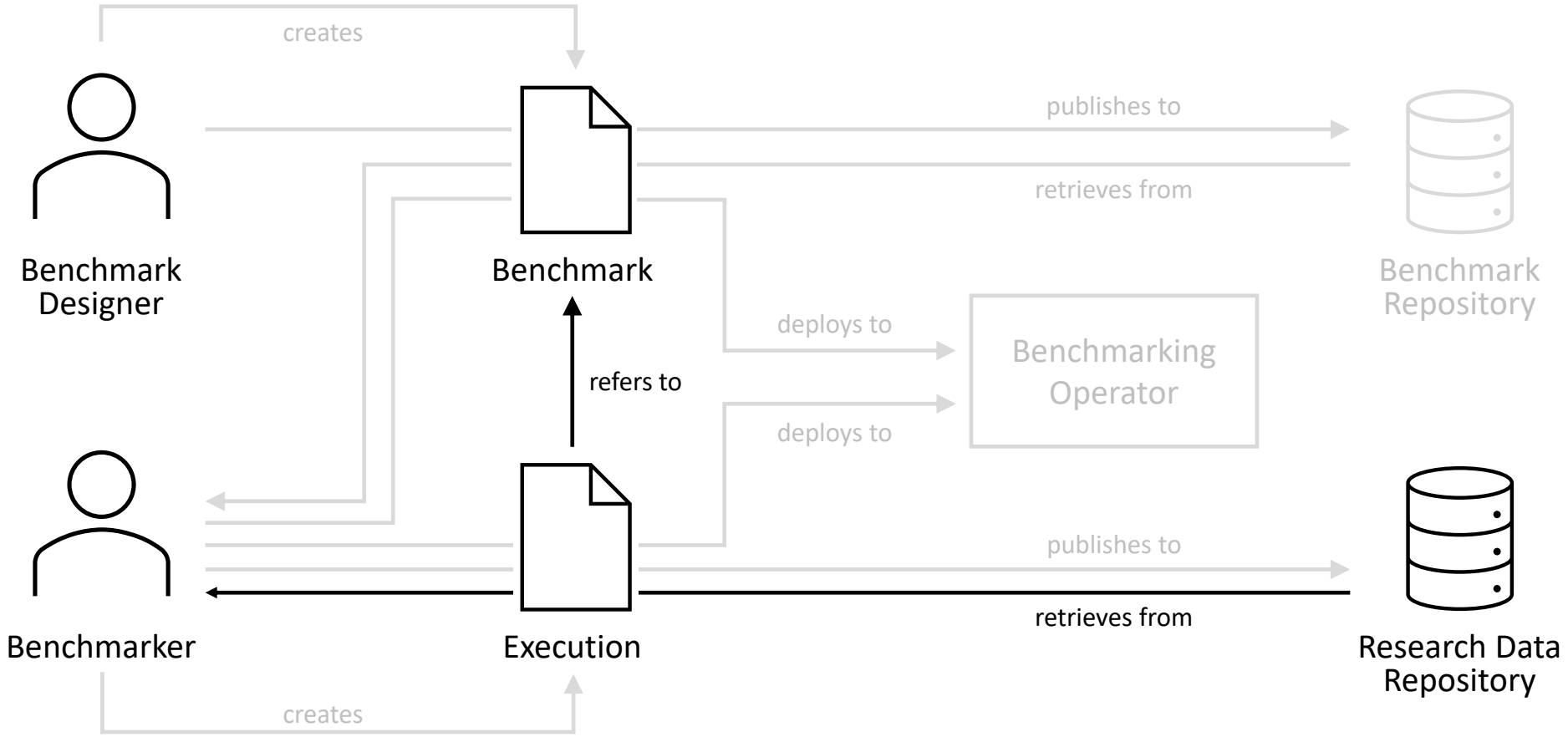
Custom Resource Definitions (CRDs) for Benchmarking



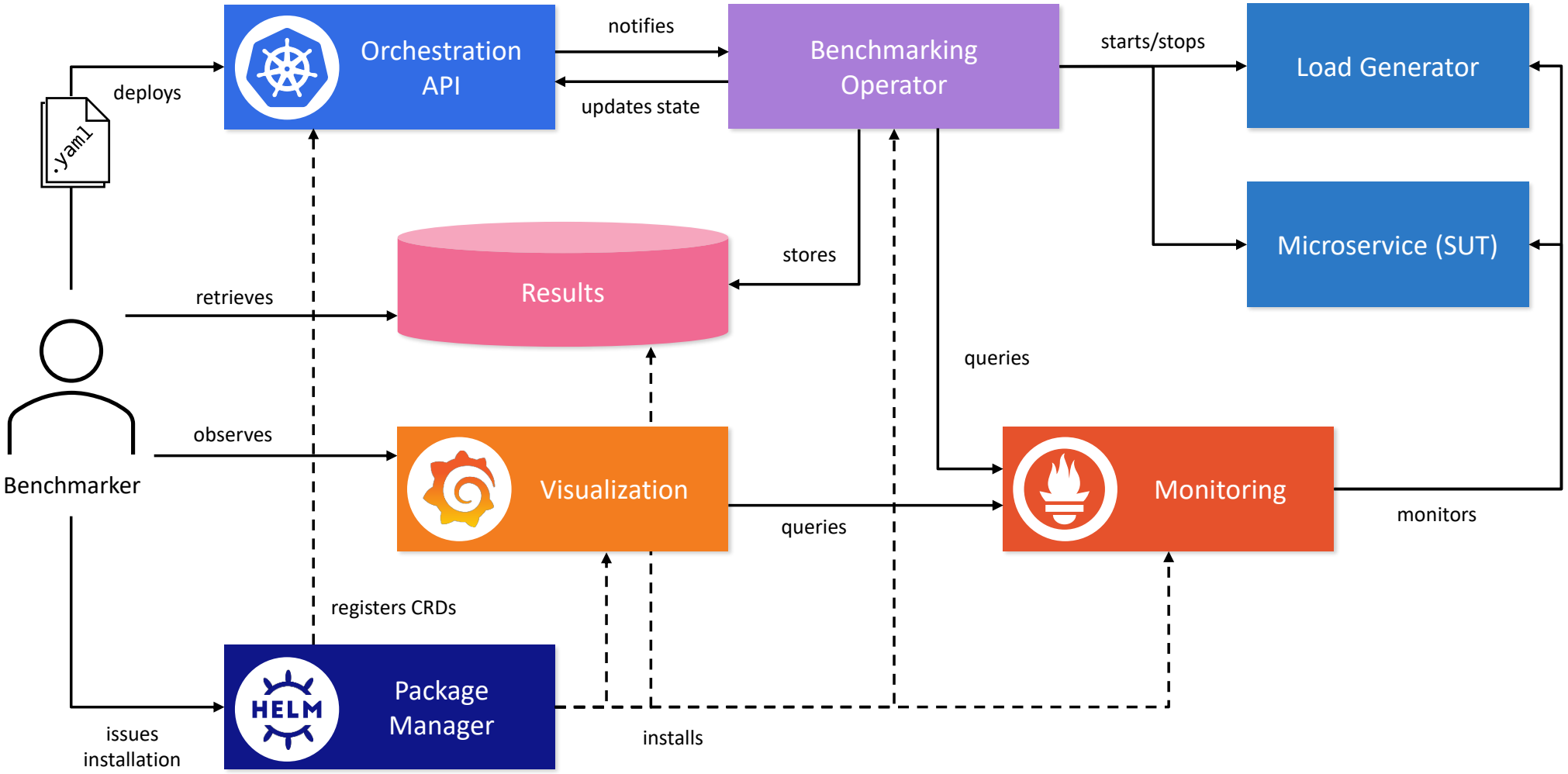
Custom Resource Definitions (CRDs) for Benchmarking



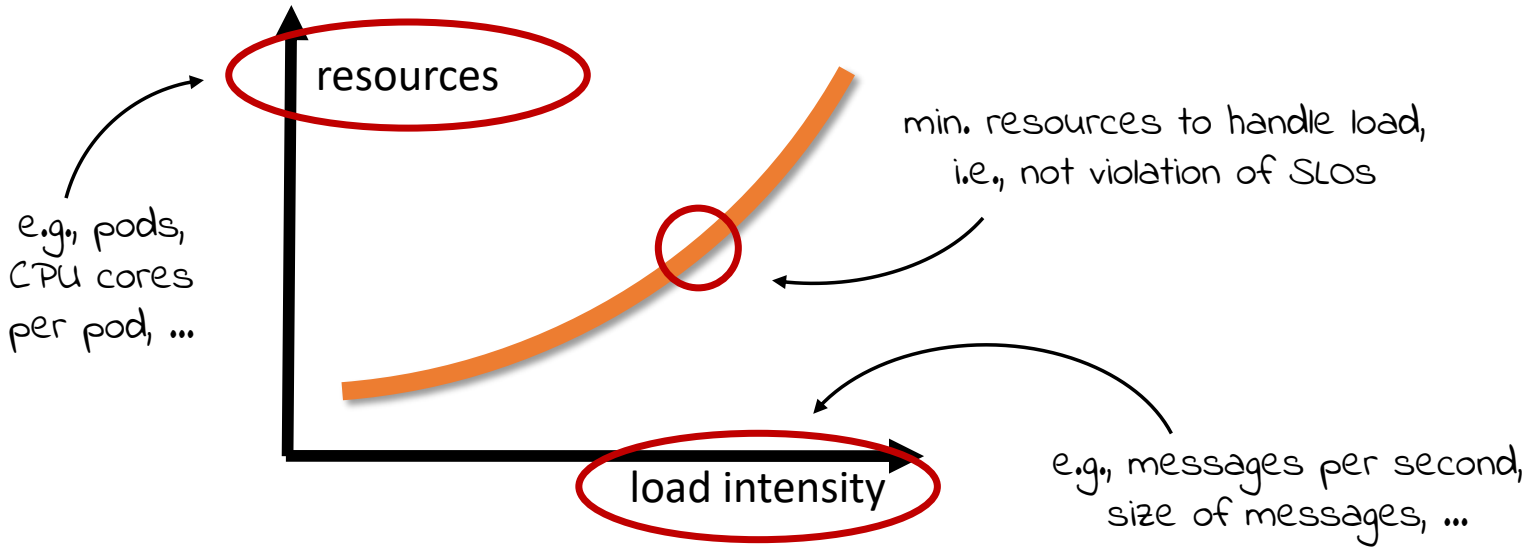
Custom Resource Definitions (CRDs) for Benchmarking



Architecture for a Benchmarking Operator



Scalability Benchmarking with Theodolite

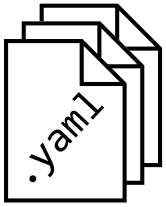


https://github.com/cau-se/theodolite

Theodolite ***Benchmarks***

```yaml
apiVersion: theodolite.com/v1
kind: benchmark
metadata:
  name: uc1-kstreams
spec:
  appResource:
    - "uc1-kstreams-deployment.yaml"
    - "aggregation-service.yaml"
    - "jmx-configmap.yaml"
    - "uc1-service-monitor.yaml"
  loadGenResource:
    - "uc1-load-generator-deployment.yaml"
    - "uc1-load-generator-service.yaml"
  resourceTypes:
    - typeName: "Instances"
      patchers:
        - type: "ReplicaPatcher"
          resource: "uc1-kstreams-deployment.yaml"
  loadTypes:
    - typeName: "NumSensors"
      patchers:
        - type: "EnvVarPatcher"
          resource: "uc1-load-generator-deployment.yaml"
          properties:
            variableName: "NUM_SENSORS"
            container: "workload-generator"
        - type: "NumSensorsLoadGeneratorReplicaPatcher"
          resource: "uc1-load-generator-deployment.yaml"
          properties:
            loadGenMaxRecords: "15000"
  kafkaConfig:
    bootstrapServer: "theodolite-cp-kafka:9092"
    topics:
      - name: "input"
        numPartitions: 40
        replicationFactor: 1
      - name: "theodolite-.*"
        removeOnly: True
```

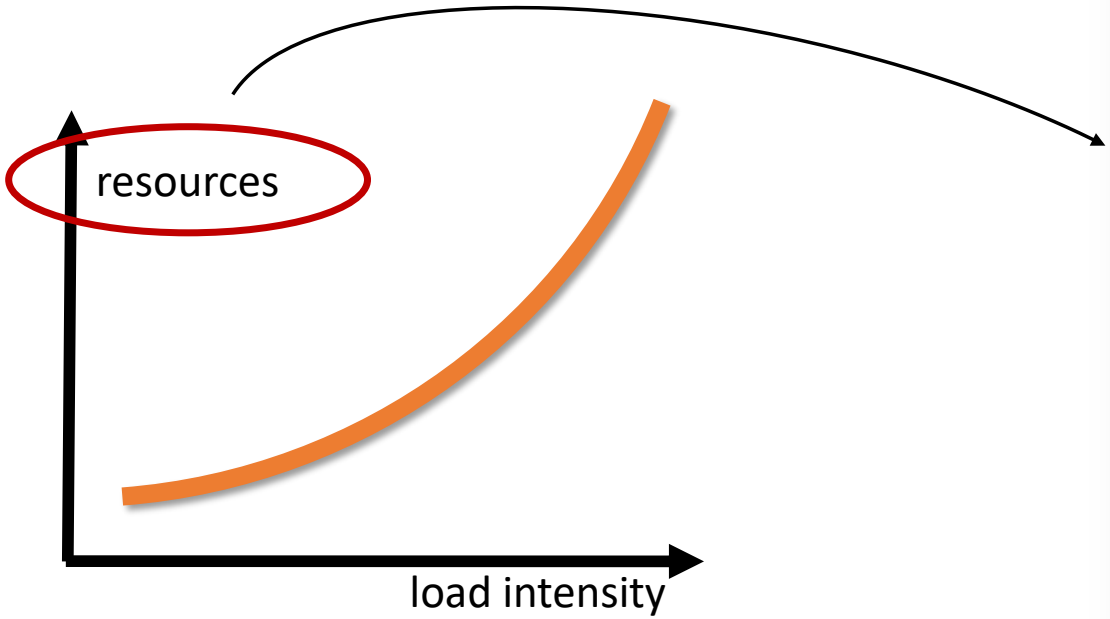
Theodolite ***Benchmarks***



Kubernetes resources for deploying the benchmark

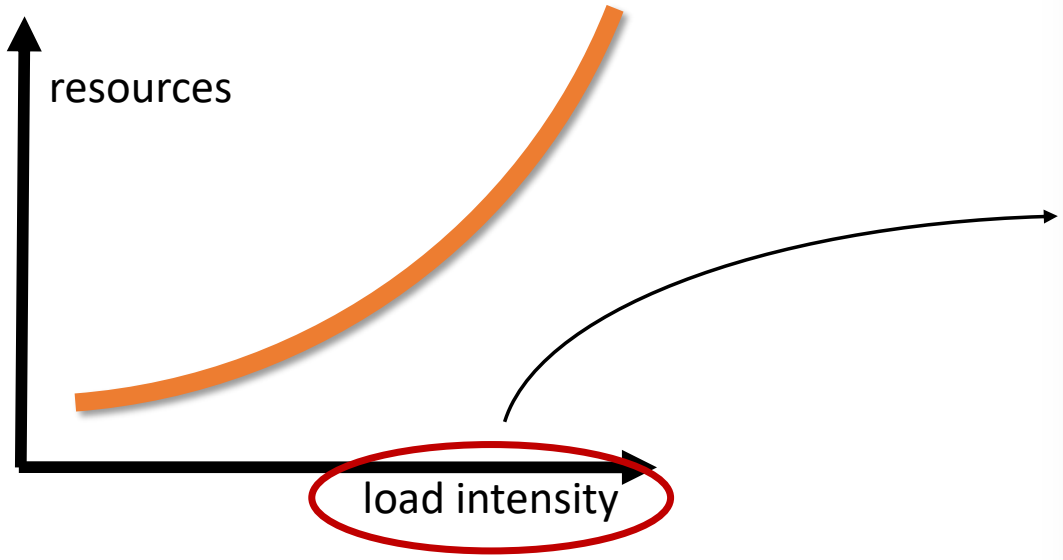
```yaml
apiVersion: theodolite.com/v1
kind: benchmark
metadata:
  name: uc1-kstreams
spec:
  appResource:
    - "uc1-kstreams-deployment.yaml"
    - "aggregation-service.yaml"
    - "jmx-configmap.yaml"
    - "uc1-service-monitor.yaml"
  loadGenResource:
    - "uc1-load-generator-deployment.yaml"
    - "uc1-load-generator-service.yaml"
  resourceTypes:
    - typeName: "Instances"
      patchers:
        - type: "ReplicaPatcher"
          resource: "uc1-kstreams-deployment.yaml"
  loadTypes:
    - typeName: "NumSensors"
      patchers:
        - type: "EnvVarPatcher"
          resource: "uc1-load-generator-deployment.yaml"
          properties:
            variableName: "NUM_SENSORS"
            container: "workload-generator"
        - type: "NumSensorsLoadGeneratorReplicaPatcher"
          resource: "uc1-load-generator-deployment.yaml"
          properties:
            loadGenMaxRecords: "15000"
  kafkaConfig:
    bootstrapServer: "theodolite-cp-kafka:9092"
    topics:
      - name: "input"
        numPartitions: 40
        replicationFactor: 1
      - name: "theodolite-.*"
        removeOnly: True
```

Theodolite ***Benchmarks***



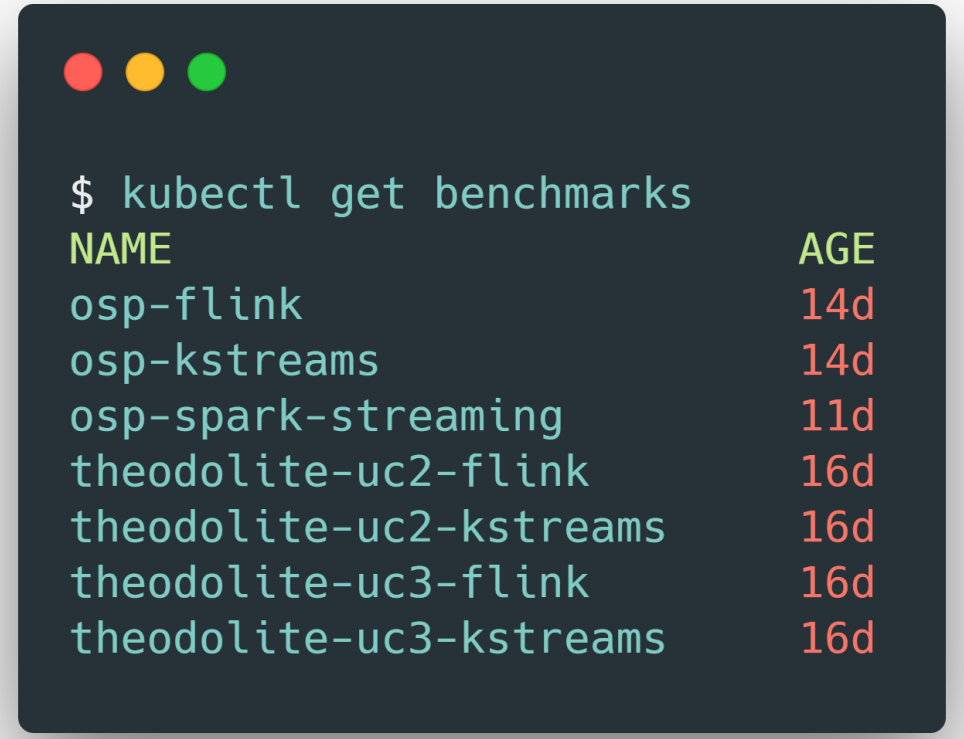

```yaml
apiVersion: theodolite.com/v1
kind: benchmark
metadata:
  name: uc1-kstreams
spec:
  appResource:
    - "uc1-kstreams-deployment.yaml"
    - "aggregation-service.yaml"
    - "jmx-configmap.yaml"
    - "uc1-service-monitor.yaml"
  loadGenResource:
    - "uc1-load-generator-deployment.yaml"
    - "uc1-load-generator-service.yaml"
  resourceTypes:
    - typeName: "Instances"
      patchers:
        - type: "ReplicaPatcher"
          resource: "uc1-kstreams-deployment.yaml"
  loadTypes:
    - typeName: "NumSensors"
      patchers:
        - type: "EnvVarPatcher"
          resource: "uc1-load-generator-deployment.yaml"
          properties:
            variableName: "NUM_SENSORS"
            container: "workload-generator"
        - type: "NumSensorsLoadGeneratorReplicaPatcher"
          resource: "uc1-load-generator-deployment.yaml"
          properties:
            loadGenMaxRecords: "15000"
  kafkaConfig:
    bootstrapServer: "theodolite-cp-kafka:9092"
    topics:
      - name: "input"
        numPartitions: 40
        replicationFactor: 1
      - name: "theodolite-.*"
        removeOnly: True
```

Theodolite ***Benchmarks***




```yaml
apiVersion: theodolite.com/v1
kind: benchmark
metadata:
  name: uc1-kstreams
spec:
  appResource:
    - "uc1-kstreams-deployment.yaml"
    - "aggregation-service.yaml"
    - "jmx-configmap.yaml"
    - "uc1-service-monitor.yaml"
  loadGenResource:
    - "uc1-load-generator-deployment.yaml"
    - "uc1-load-generator-service.yaml"
  resourceTypes:
    - typeName: "Instances"
      patchers:
        - type: "ReplicaPatcher"
          resource: "uc1-kstreams-deployment.yaml"
  loadTypes:
    - typeName: "NumSensors"
      patchers:
        - type: "EnvVarPatcher"
          resource: "uc1-load-generator-deployment.yaml"
          properties:
            variableName: "NUM_SENSORS"
            container: "workload-generator"
        - type: "NumSensorsLoadGeneratorReplicaPatcher"
          resource: "uc1-load-generator-deployment.yaml"
          properties:
            loadGenMaxRecords: "15000"
  kafkaConfig:
    bootstrapServer: "theodolite-cp-kafka:9092"
    topics:
      - name: "input"
        numPartitions: 40
        replicationFactor: 1
      - name: "theodolite-.*"
        removeOnly: True
```

Theodolite ***Benchmarks***

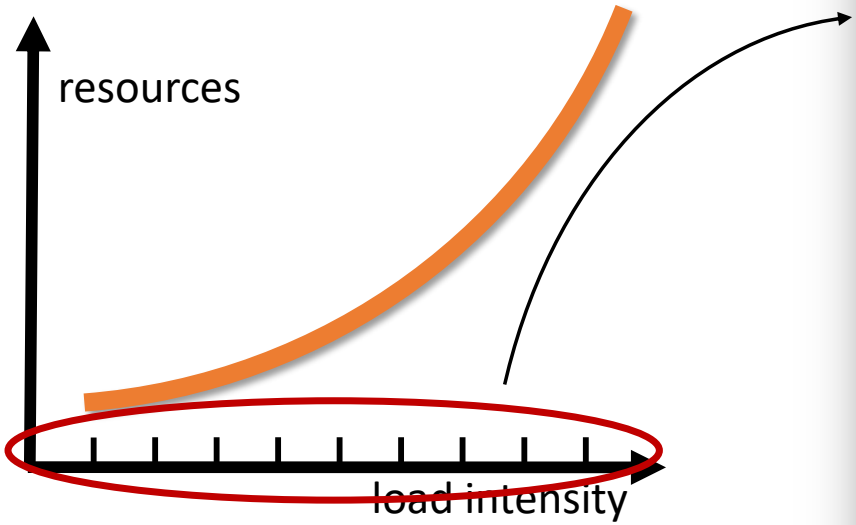


Theodolite ***Executions***

Benchmark

```yaml
apiVersion: theodolite.com/v1
kind: execution
metadata:
  name: theodolite-example-execution
spec:
  benchmark: "uc1-kstreams"
  load:
    loadType: "NumSensors"
    loadValues: [25000, 50000, 75000, 100000, 125000, 150000]
  resources:
    resourceType: "Instances"
    resourceValues: [1, 2, 3, 4, 5]
  slos:
    - sloType: "lag trend"
      prometheusUrl: "http://prometheus-operated:9090"
      offset: 0
      properties:
        threshold: 2000
        externalSloUrl: "http://localhost:80/evaluate-slope"
        warmup: 60 # in seconds
  execution:
    strategy: "LinearSearch"
    duration: 300 # in seconds
    repetitions: 1
    loadGenerationDelay: 30 # in seconds
    restrictions:
      - "LowerBound"
  configOverrides: []
```

Theodolite ***Executions***




```yaml
apiVersion: theodolite.com/v1
kind: execution
metadata:
  name: theodolite-example-execution
spec:
  benchmark: "uc1-kstreams"
  load:
    loadType: "NumSensors"
    loadValues: [25000, 50000, 75000, 100000, 125000, 150000]
  resources:
    resourceType: "Instances"
    resourceValues: [1, 2, 3, 4, 5]
  slos:
    - sloType: "lag trend"
      prometheusUrl: "http://prometheus-operated:9090"
      offset: 0
      properties:
        threshold: 2000
        externalSloUrl: "http://localhost:80/evaluate-slope"
        warmup: 60 # in seconds
  execution:
    strategy: "LinearSearch"
    duration: 300 # in seconds
    repetitions: 1
    loadGenerationDelay: 30 # in seconds
    restrictions:
      - "LowerBound"
  configOverrides: []
```

Theodolite ***Executions***

Prometheus

pandas

SLO evaluation

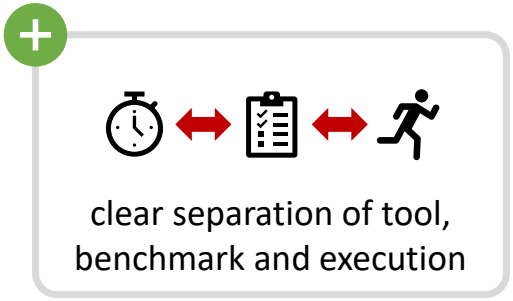
```yaml
apiVersion: theodolite.com/v1
kind: execution
metadata:
  name: theodolite-example-execution
spec:
  benchmark: "uc1-kstreams"
  load:
    loadType: "NumSensors"
    loadValues: [25000, 50000, 75000, 100000, 125000, 150000]
  resources:
    resourceType: "Instances"
    resourceValues: [1, 2, 3, 4, 5]
  slos:
    - sloType: "lag trend"
      prometheusUrl: "http://prometheus-operated:9090"
      offset: 0
      properties:
        threshold: 2000
        externalSloUrl: "http://localhost:80/evaluate-slope"
        warmup: 60 # in seconds
  execution:
    strategy: "LinearSearch"
    duration: 300 # in seconds
    repetitions: 1
    loadGenerationDelay: 30 # in seconds
    restrictions:
      - "LowerBound"
  configOverrides: []
```

Theodolite ***Executions***

```
$ kubectl get executions
NAME                                STATUS     DURATION   AGE
osp-flink-execution                 PENDING    -          6s
osp-flink-execution-small           RUNNING    2m         2m52s
osp-kstreams-default                FINISHED   11h        10d
osp-kstreams-execution              FINISHED   42m        14d
osp-spark-streaming-execution       FINISHED   8m         11d
uc3-flink-template                  FINISHED   18m        11d
uc3-kstreams-default                FINISHED   10h        10d
uc3-kstreams-default-short          FINISHED   3h         10d
uc3-kstreams-scotty                 FINISHED   4h         10d
uc3-kstreams-scotty-short           FINISHED   2h         10d
```

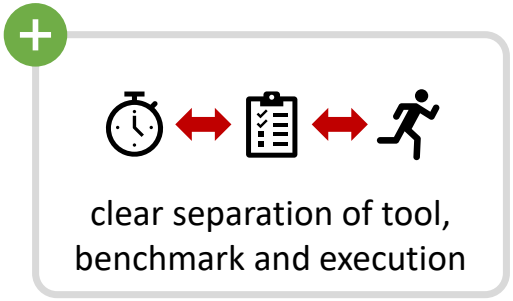
Summary: An Operator for Cloud-native Benchmarks

Summary: An Operator for Cloud-native Benchmarks

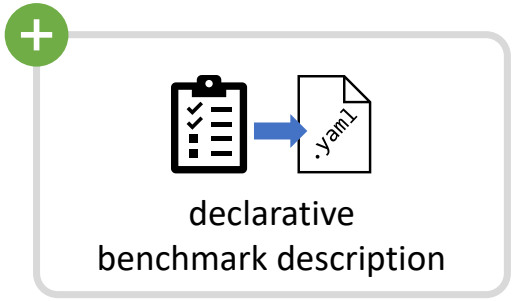


clear separation of tool, benchmark and execution

Summary: An Operator for Cloud-native Benchmarks

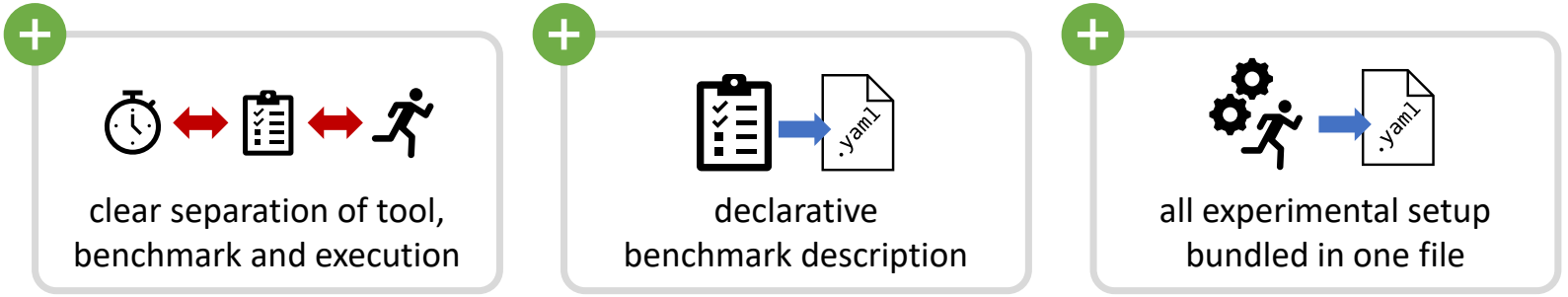


clear separation of tool, benchmark and execution



declarative benchmark description

Summary: An Operator for Cloud-native Benchmarks



clear separation of tool, benchmark and execution

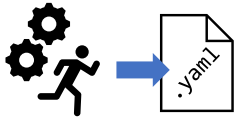
declarative benchmark description

all experimental setup bundled in one file

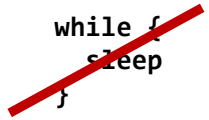
Summary: An Operator for Cloud-native Benchmarks

clear separation of tool, benchmark and execution

declarative benchmark description



all experimental setup bundled in one file

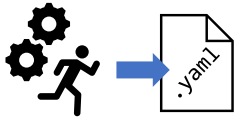


no Kubernetes error handling etc. in benchmark

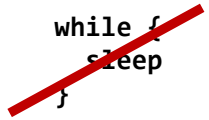
Summary: An Operator for Cloud-native Benchmarks

clear separation of tool, benchmark and execution

declarative benchmark description



all experimental setup bundled in one file



no Kubernetes error handling etc. in benchmark

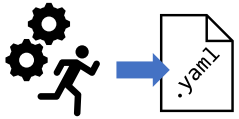
```
> kubectl _
```

utilizing established Kubernetes tools

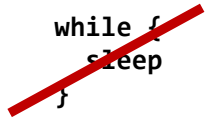
Summary: An Operator for Cloud-native Benchmarks

clear separation of tool, benchmark and execution

declarative benchmark description



all experimental setup bundled in one file



no Kubernetes error handling etc. in benchmark

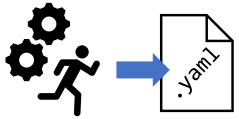
utilizing established Kubernetes tools

built-in schema validation by Kubernetes

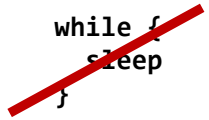
Summary: An Operator for Cloud-native Benchmarks

clear separation of tool, benchmark and execution

declarative benchmark description



all experimental setup bundled in one file



no Kubernetes error handling etc. in benchmark

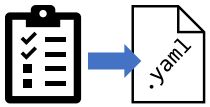
utilizing established Kubernetes tools

built-in schema validation by Kubernetes

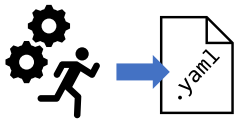
enabling continuous benchmarking with GitOps

Summary: An Operator for Cloud-native Benchmarks

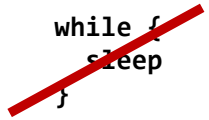
clear separation of tool, benchmark and execution



declarative benchmark description



all experimental setup bundled in one file



no Kubernetes error handling etc. in benchmark

utilizing established Kubernetes tools

built-in schema validation by Kubernetes

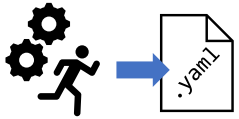
enabling continuous benchmarking with GitOps

increased implementation effort

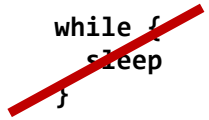
Summary: An Operator for Cloud-native Benchmarks

clear separation of tool, benchmark and execution

declarative benchmark description



all experimental setup bundled in one file



no Kubernetes error handling etc. in benchmark

utilizing established Kubernetes tools

built-in schema validation by Kubernetes

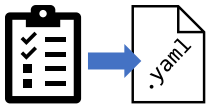
enabling continuous benchmarking with GitOps

increased implementation effort

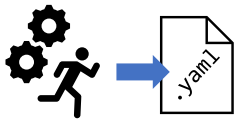
However:

Summary: An Operator for Cloud-native Benchmarks

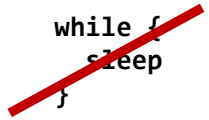
clear separation of tool, benchmark and execution



declarative benchmark description



all experimental setup bundled in one file



no Kubernetes error handling etc. in benchmark

> kubectl _

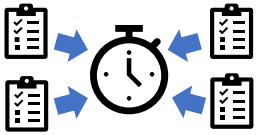
utilizing established Kubernetes tools

built-in schema validation by Kubernetes

enabling continuous benchmarking with GitOps

increased implementation effort

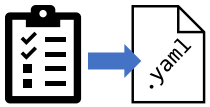
However:



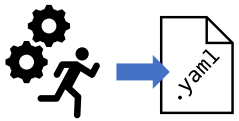
new benchmarks may utilize existing operators

Summary: An Operator for Cloud-native Benchmarks

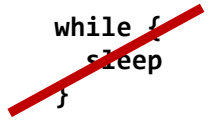
clear separation of tool, benchmark and execution



declarative benchmark description



all experimental setup bundled in one file



no Kubernetes error handling etc. in benchmark

> kubectl _

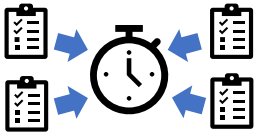
utilizing established Kubernetes tools

built-in schema validation by Kubernetes

enabling continuous benchmarking with GitOps

increased implementation effort

However:

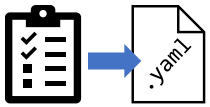


new benchmarks may utilize existing operators

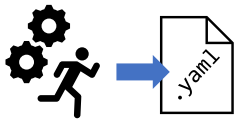
lots of tooling available for creating new operators

Summary: An Operator for Cloud-native Benchmarks

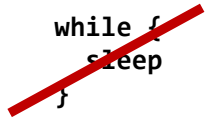
clear separation of tool, benchmark and execution



declarative benchmark description



all experimental setup bundled in one file



no Kubernetes error handling etc. in benchmark

> kubectl _

utilizing established Kubernetes tools

OPENAPI INITIATIVE

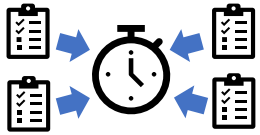
built-in schema validation by Kubernetes

argo flux

enabling continuous benchmarking with GitOps

increased implementation effort

However:



new benchmarks may utilize existing operators

OPERATOR FRAMEWORK

fabric8

lots of tooling available for creating new operators

https://github.com/cau-se/theodolite