jq + kubectl for SRE

Incident-oriented JSON queries for Kubernetes resources

v1.1.0

Pocketbook Series

by Luca Sepe

LS[71]

Index

Pending and Unschedulable Pods

When to use

understand why pods do not start

Pending pods with scheduling reason/message

```
kubectl get pods -A -o json \
| jq '.items[]
      | select(.status.phase == "Pending")
      | {
          ns: .metadata.namespace,
          pod: .metadata.name,
          reason: (.status.conditions[]? |
    select(.type=="PodScheduled") | .reason),
          message: (.status.conditions[]? |
    select(.type=="PodScheduled") | .message)
        }'
```

Pods stuck terminating

When to use

detect rollout drains or finalizers blocking shutdown

```
# deletionTimestamp present means terminating path started
kubectl get pods -A -o json \
| jq '.items[]
      | select(.metadata.deletionTimestamp != null)
      | {
          ns: .metadata.namespace,
          pod: .metadata.name,
          deleting_since: .metadata.deletionTimestamp,
          grace: (.spec.terminationGracePeriodSeconds //
    "default")
        }'
```

Node Health and Scheduling

When to use

isolate node-level pressure affecting pod placement

Node readiness + pressure conditions

```
kubectl get nodes -o json \
| jq '.items[]
      | {
          node: .metadata.name,
          ready: ([.status.conditions[] |
     select(.type=="Ready") | .status][0] // "Unknown"),
          memPressure: ([.status.conditions[] |
     select(.type=="MemoryPressure") | .status][0] // "Unknown"),
          diskPressure: ([.status.conditions[] |
     select(.type=="DiskPressure") | .status][0] // "Unknown"),
          pidPressure: ([.status.conditions[] |
     select(.type=="PIDPressure") | .status][0] // "Unknown")
        }'
```

Unschedulable nodes

```
kubectl get nodes -o json \
| jq '.items[] | select(.spec.unschedulable == true) |
     .metadata.name'
```

Workload Rollout and Drift

When to use

check if rollouts converged to desired replicas

Deployments not fully available

```
kubectl get deploy -A -o json \
| jq '.items[]
      | {
          ns: .metadata.namespace,
          deploy: .metadata.name,
          desired: (.spec.replicas // 0),
          available: (.status.availableReplicas // 0),
          updated: (.status.updatedReplicas // 0)
        }
      | select(.available < .desired or .updated < .desired)'
```

StatefulSets not ready

```
kubectl get sts -A -o json \
| jq '.items[]
      | {
          ns: .metadata.namespace,
          sts: .metadata.name,
          desired: (.spec.replicas // 0),
          ready: (.status.readyReplicas // 0)
        }
      | select(.ready < .desired)'
```

Events Triage

When to use

move from symptom to probable root cause with event reasons

Warning events (newest first)

```
kubectl get events -A --field-selector type=Warning -o json \
| jq '.items
      | sort_by(.lastTimestamp // .eventTime //
     .metadata.creationTimestamp)
      | reverse
      | .[]
      | {
          ns: .metadata.namespace,
          time: (.lastTimestamp // .eventTime //
     .metadata.creationTimestamp),
          reason: .reason,
          object: (.involvedObject.kind + "/" +
     .involvedObject.name),
          message: .message
        }'
```

Top warning reasons by count

```
kubectl get events -A --field-selector type=Warning -o json \
| jq '.items
      | group_by(.reason)
      | map({reason: .[0].reason, count: length})
      | sort_by(.count)
      | reverse
      | .[:15]'
```