
# Don’t Catch Feelings, Catch Issues With Kuberhealthy

Joshulyne Park

Shilla Saebi


# Kuberhealthy: An Operator for Synthetic Monitoring on Kubernetes

Joshulyne Park

Shilla Saebi

# OPEN SOURCE AT COMCAST

OPEN SOURCE
AT COMCAST

STORYTIME

OPEN SOURCE
AT COMCAST

STORYTIME

KUBERHEALTHY
2.0.0

OPEN SOURCE AT COMCAST

STORYTIME

KUBERHEALTHY 2.0.0

DEMO

OPEN SOURCE AT COMCAST

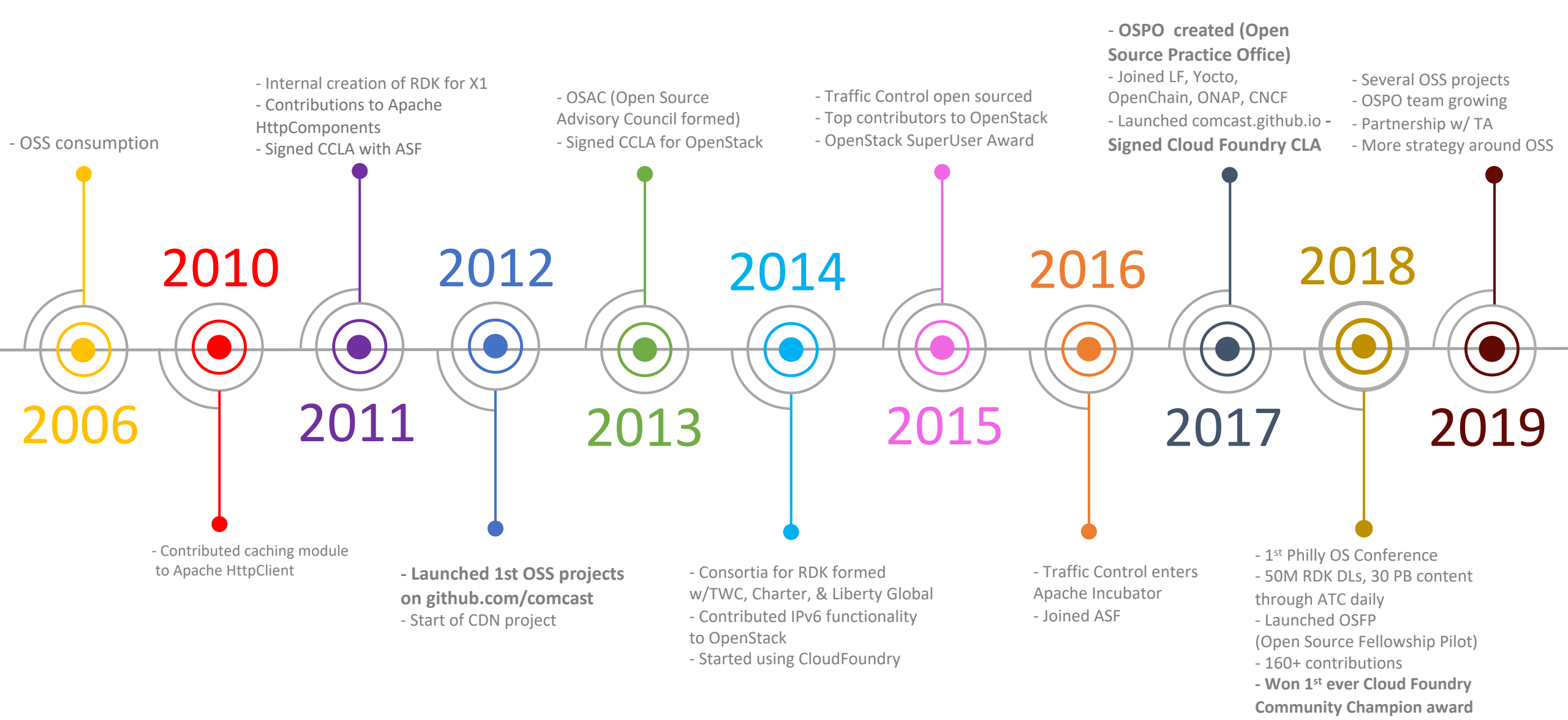
STORYTIME

KUBERHEALTHY 2.0.0

DEMO

CONTRIBUTE

# Comcast's Open Source Journey




Opened in 2017

Support 8k+ developers

200+ repos on GitHub

7-person team

# Open Source Program Office


Kubernetes Operator

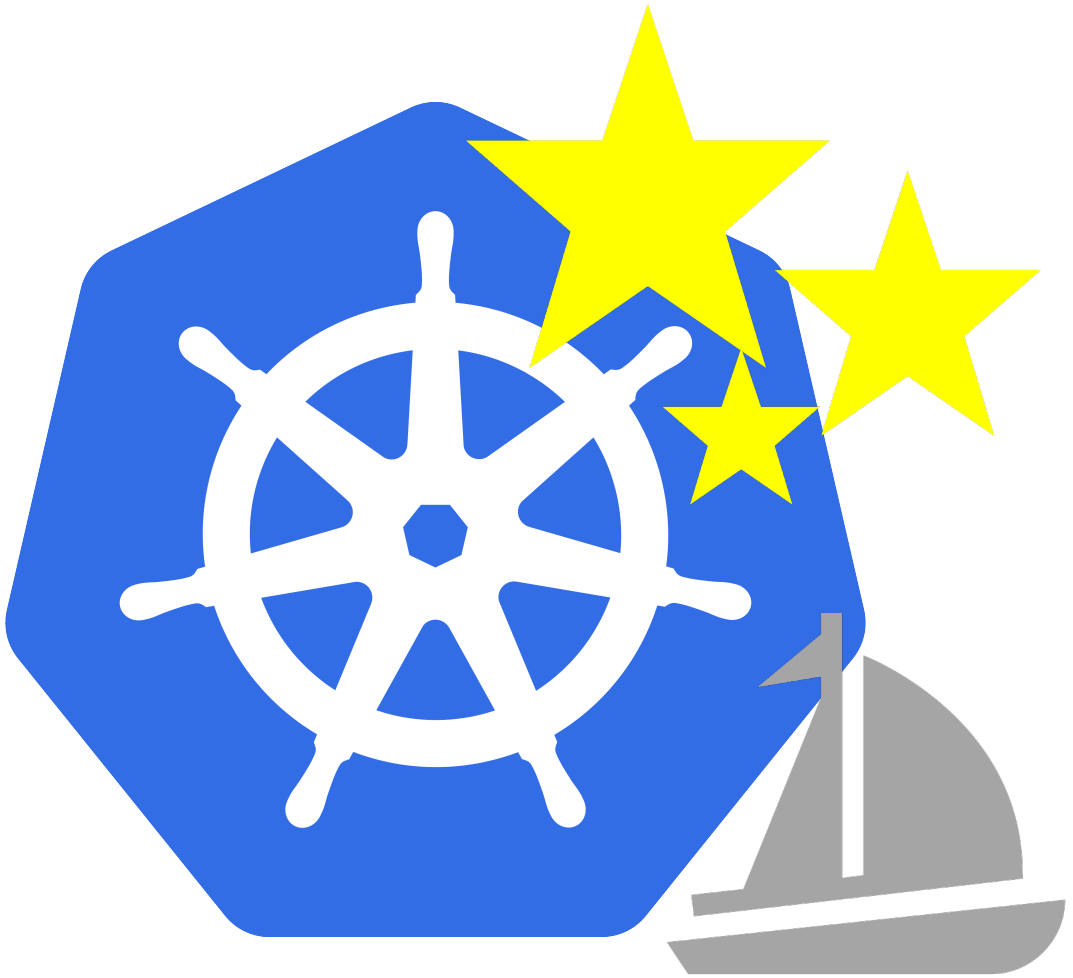

Kubernetes Operator


Application Developer


Kubernetes Operator


Kubernetes Operator


Kubernetes Operator




Application Developer

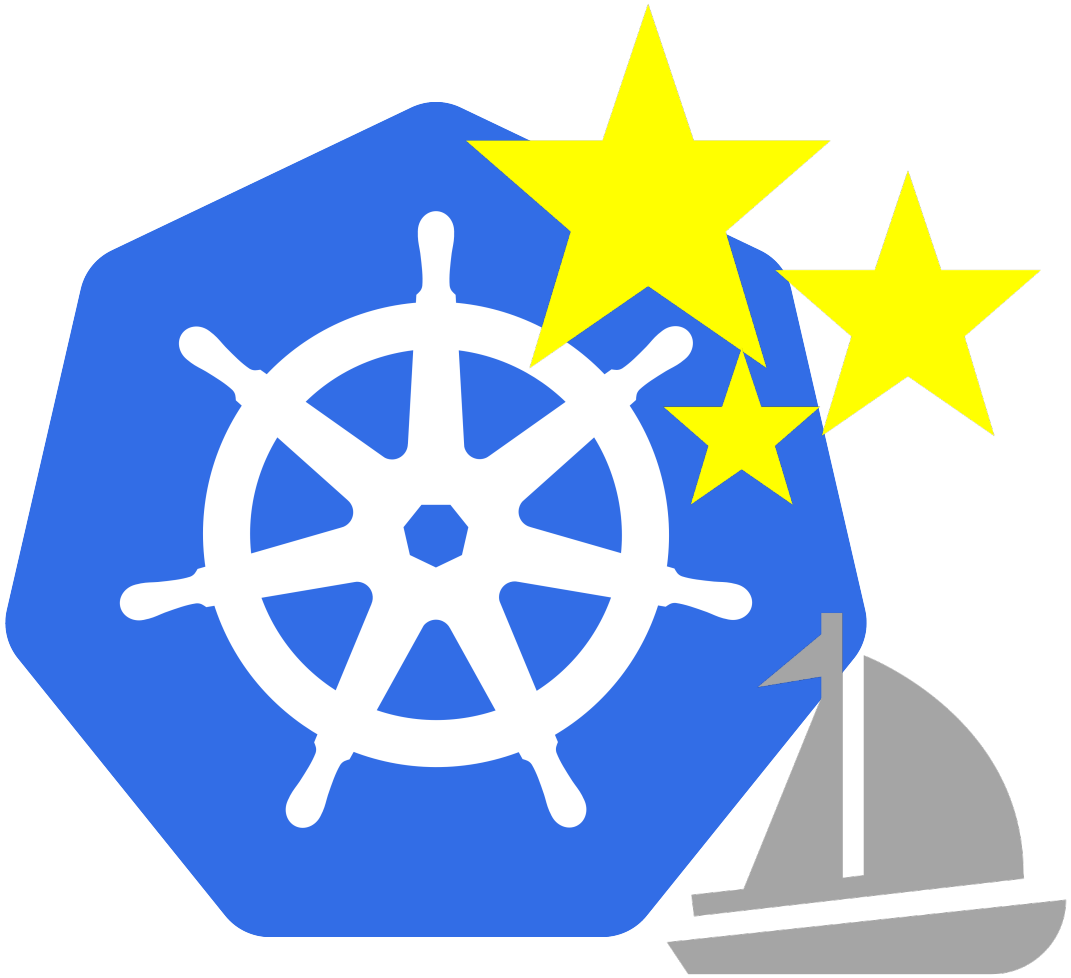

Application Developer


Kubernetes Operator

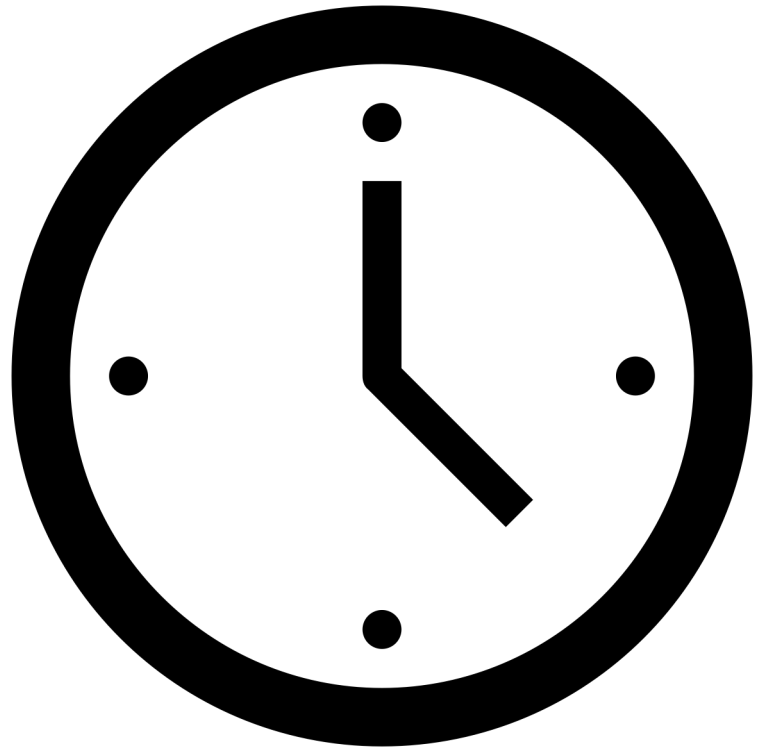

Kubernetes Operator


Application Developer


Application Developer




Application Developer




KIM

ContainerCreating


JAKE


JAKE


JAKE


JAKE

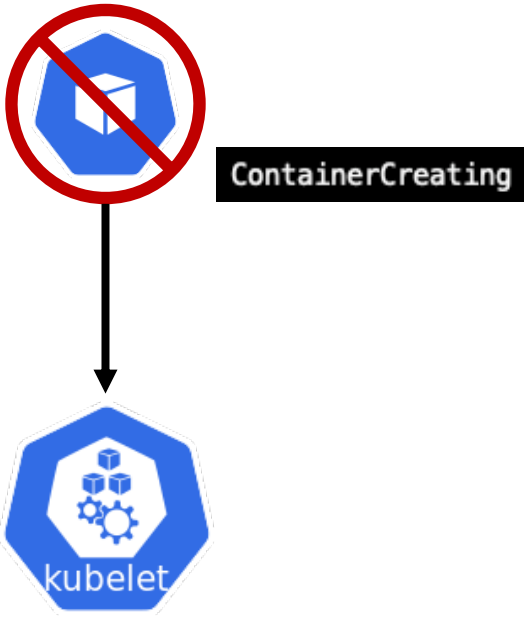

node



ContainerCreating

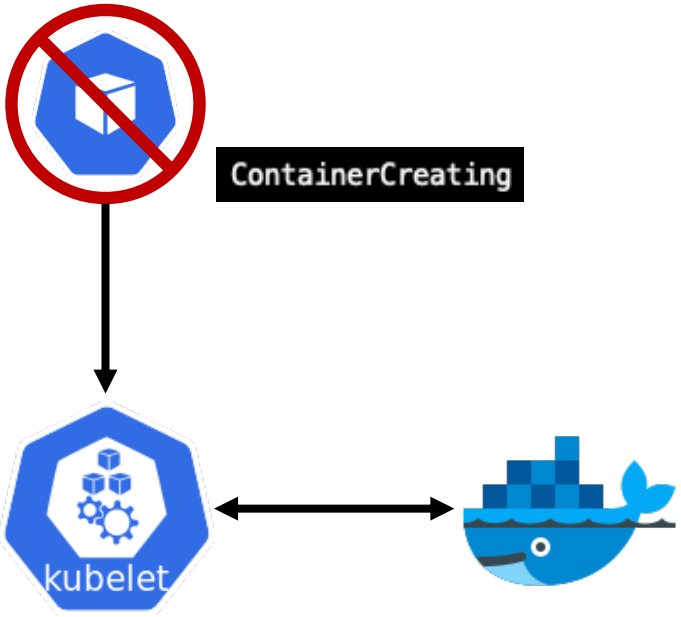

JAKE


node


ContainerCreating
kubelet


JAKE

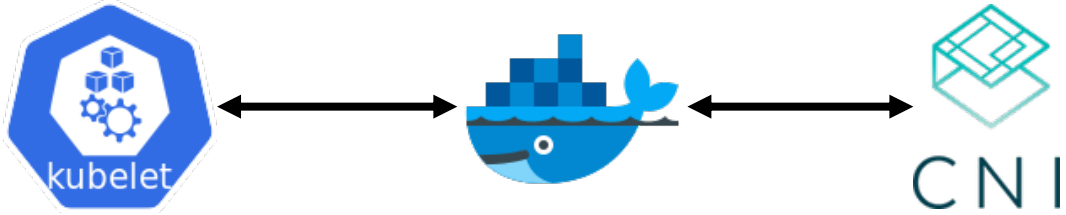

node


ContainerCreating
kubelet


JAKE

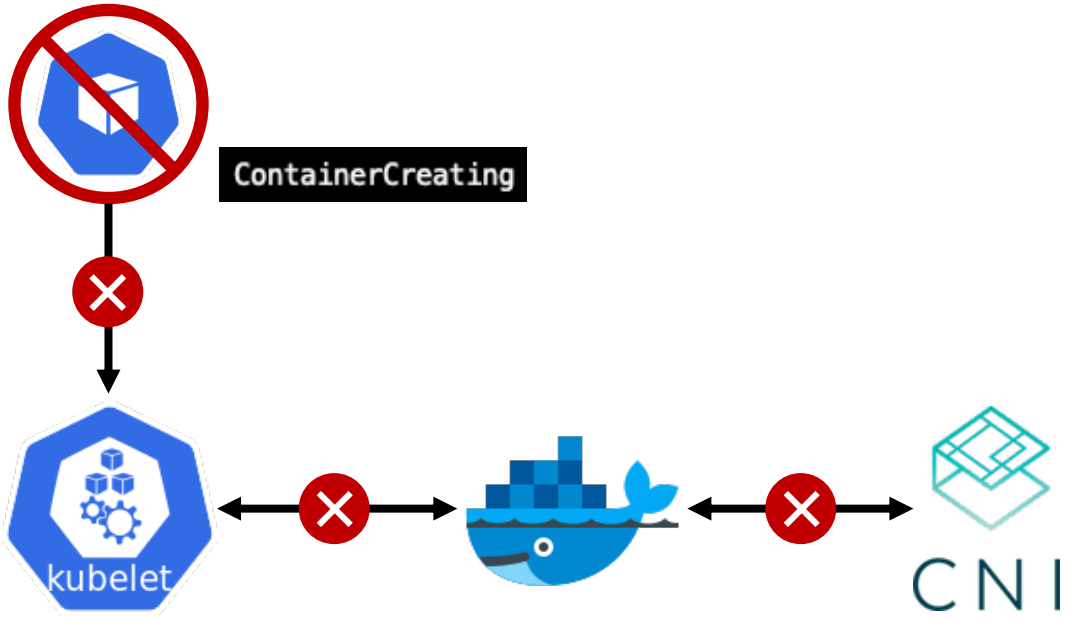

node


ContainerCreating


kubelet
CNI


JAKE


node


ContainerCreating
kubelet
CNI

Pending

Terminating


# Related Issues

CNI communication failures


# Related Issues

CNI communication failures


Docker / Kubelet crashes or restarts


# Related Issues

CNI communication failures


Docker / Kubelet crashes or restarts

Disk provisioning failures


# Related Issues

CNI communication failures


Docker / Kubelet crashes or restarts

Disk provisioning failures


# Related Issues

High I/O wait times

CNI communication failures


Docker / Kubelet crashes or restarts

Disk provisioning failures


# Related Issues

Container runtime errors

High I/O wait times

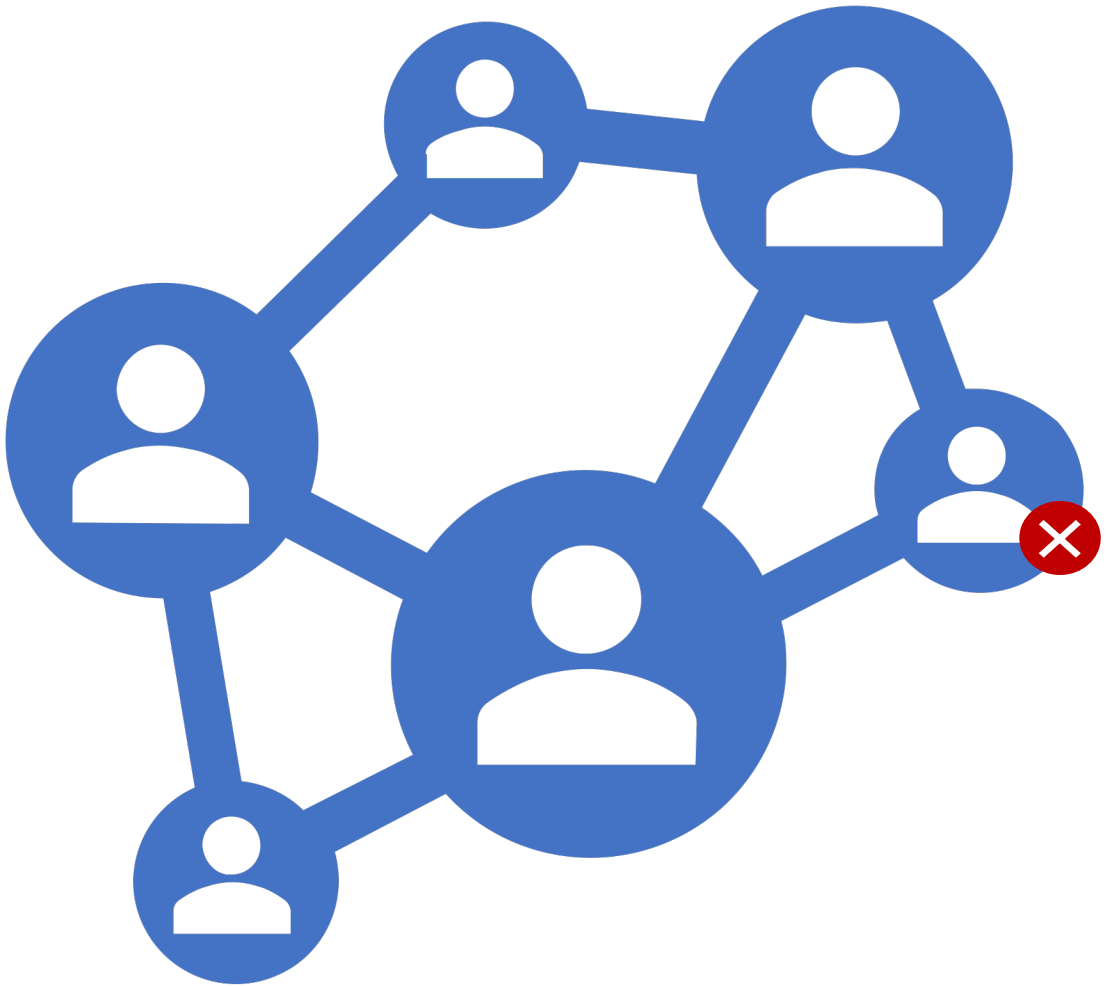
Kubernetes system services remaining technically "healthy" while their underlying pods are crashing too much

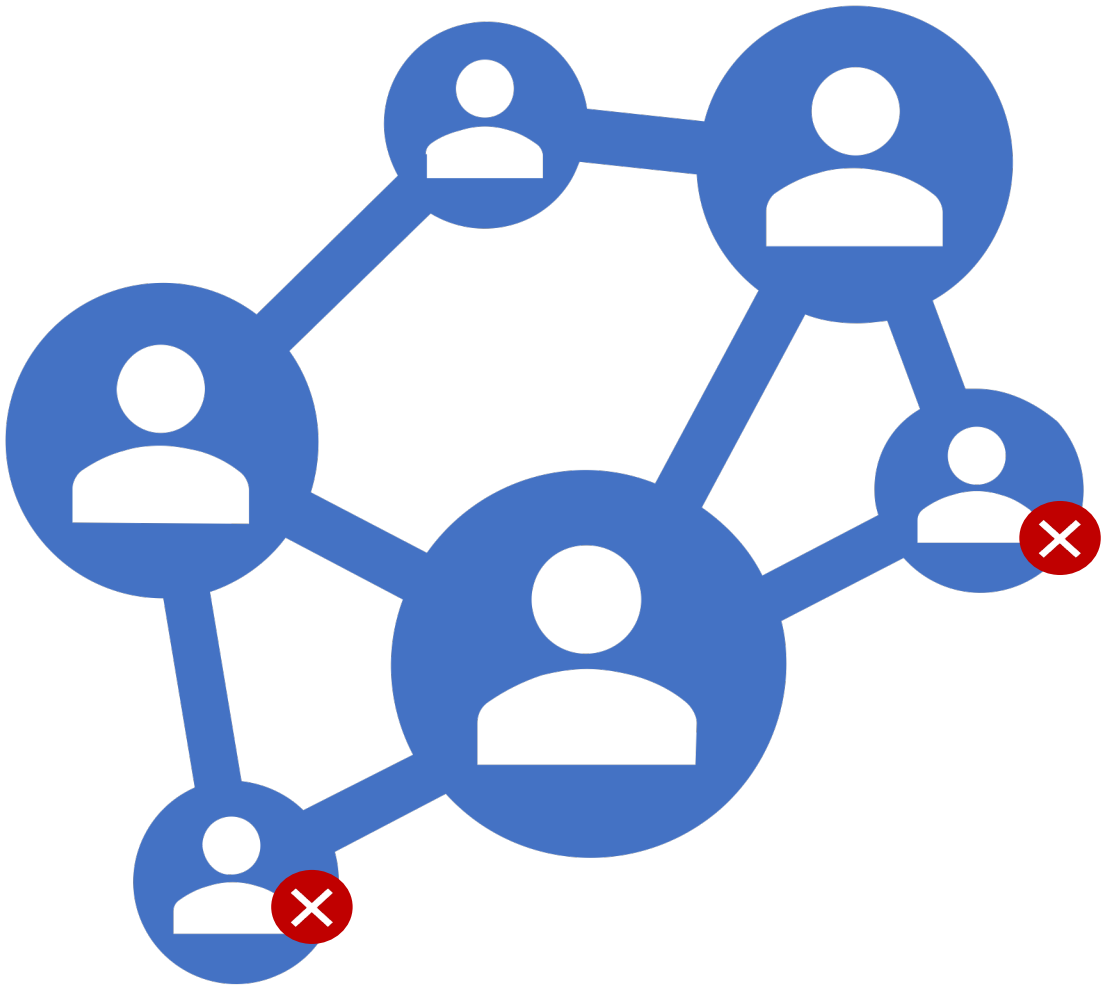


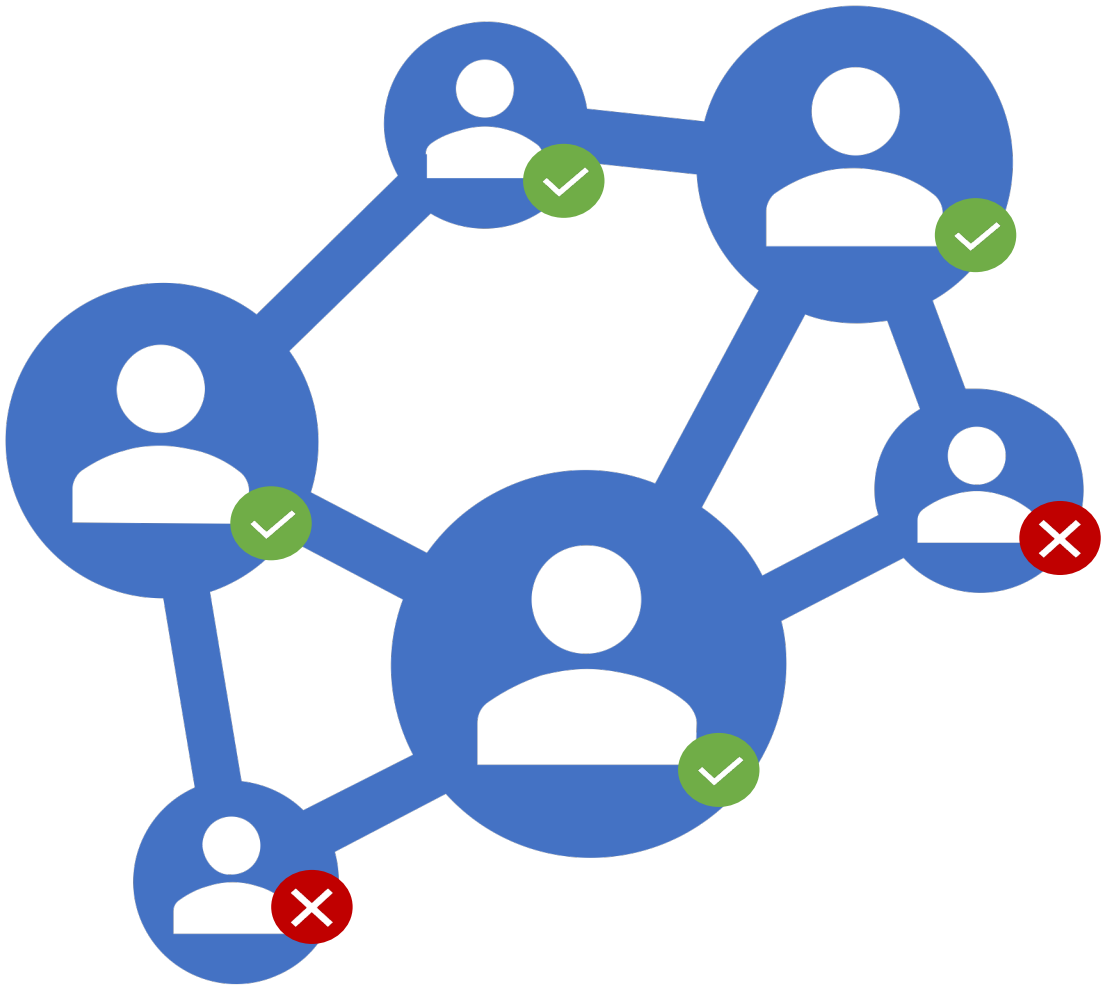
Kubernetes system services remaining technically "healthy" while their underlying pods are crashing too much



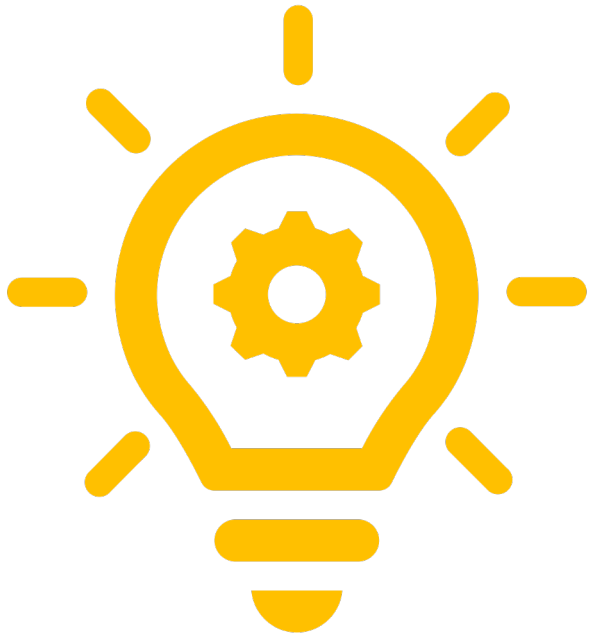
Kubernetes system services remaining technically "healthy" while their underlying pods are crashing too much



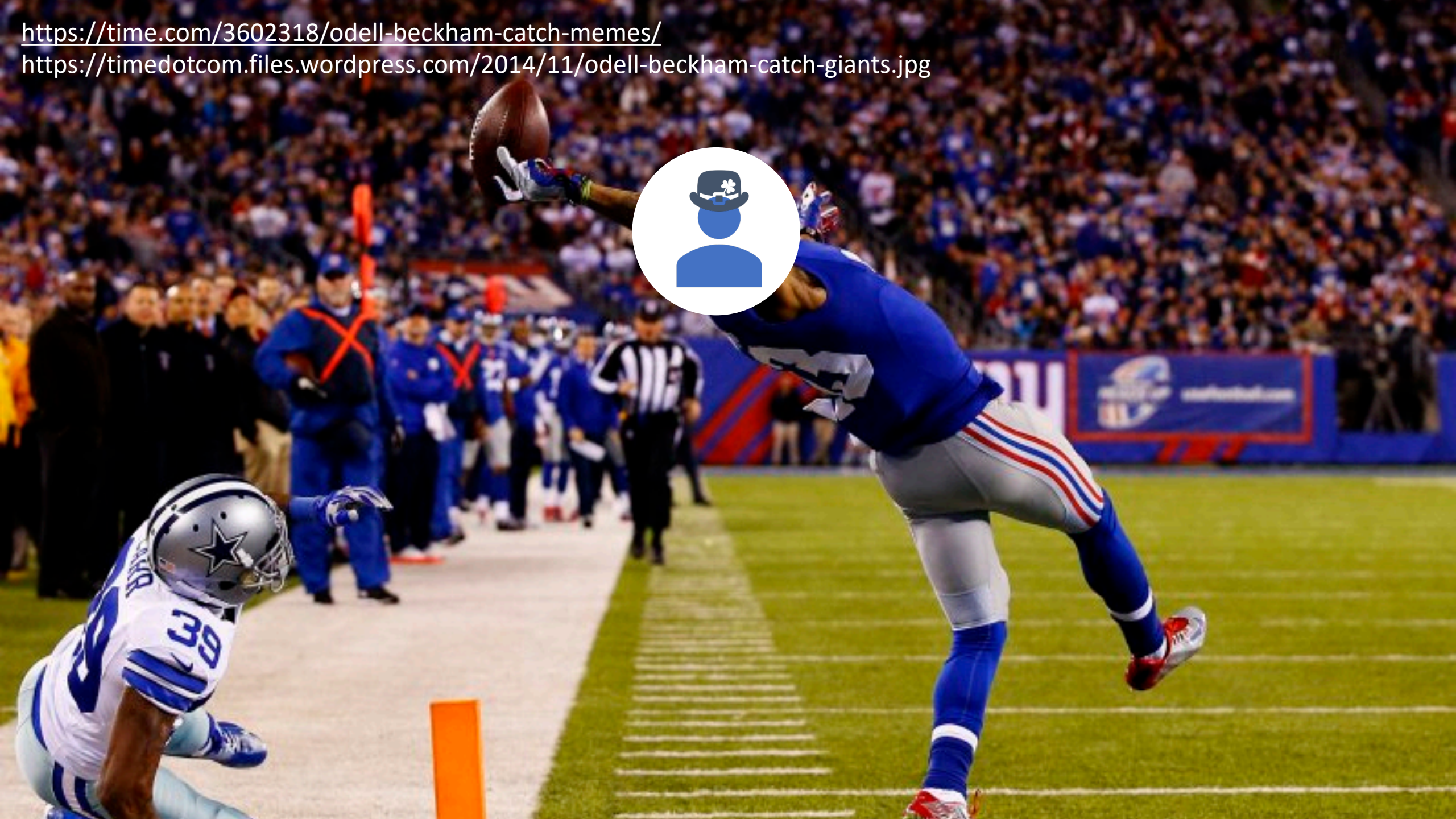




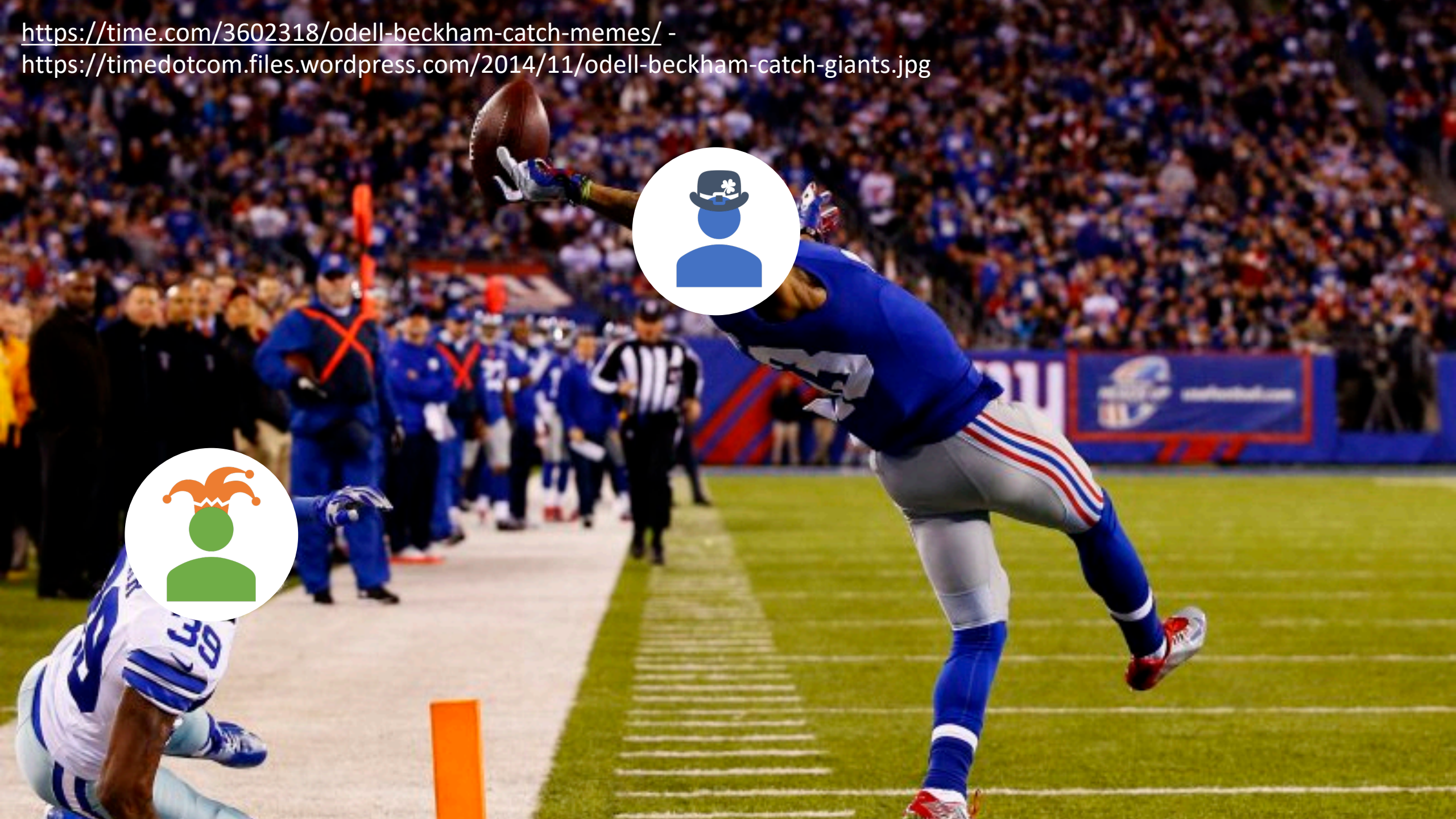

JAKE



https://time.com/3602318/odell-beckham-catch-memes/
https://timedotcom.files.wordpress.com/2014/11/odell-beckham-catch-giants.jpg



https://time.com/3602318/odell-beckham-catch-memes/ -
https://timedotcom.files.wordpress.com/2014/11/odell-beckham-catch-giants.jpg

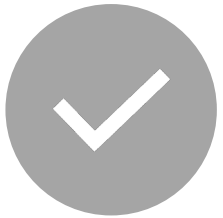



# Daemonset Check


# Deployment Check

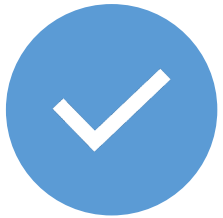

# DNS Status Check

DAEMONSET CHECK



DEPLOYMENT CHECK

POD RESTARTS CHECK



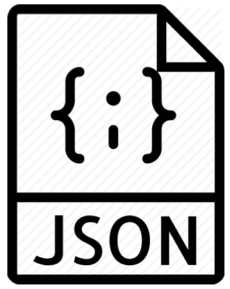
POD STATUS

DNS STATUS CHECK

AND MORE!

```
{
 "OK": true,
  "Errors": [],
  "CheckDetails": {
    "dns-status-check-external": {
      "OK": true,
      "Errors": [],
      "Namespace": "kuberhealthy",
      "LastRun": "2019-10-30T17:45:02.199703616Z",
      "AuthoritativePod": "kuberhealthy-5bfbb7f8b-6hkgs"
    },
    "dns-status-check-internal": {
      "OK": true,
      "Errors": [],
      "Namespace": "kuberhealthy",
      "LastRun": "2019-10-30T17:45:00.924306952Z",
      "AuthoritativePod": "kuberhealthy-5bfbb7f8b-6hkgs"
    },
    "kh-daemonset-check": {
      "OK": true,
      "Errors": [],
      "Namespace": "kuberhealthy",
      "LastRun": "2019-10-30T17:31:29.505816573Z",
      "AuthoritativePod": "kuberhealthy-5bfbb7f8b-6hkgs"
    },
    "kh-deployment-check": {
      "OK": true,
      "Errors": [],
      "Namespace": "kuberhealthy",
      "LastRun": "2019-10-30T17:32:21.993233082Z",
      "AuthoritativePod": "kuberhealthy-5bfbb7f8b-6hkgs"
    },
    "pod-restarts-check": {
      "OK": true,
      "Errors": [],
      "Namespace": "kuberhealthy",
      "LastRun": "2019-10-30T17:44:56.718123421Z",
      "AuthoritativePod": "kuberhealthy-5bfbb7f8b-t9kfs"
    }
  },
  "CurrentMaster": "kuberhealthy-5bfbb7f8b-6hkgs"
}
```


# Check Status



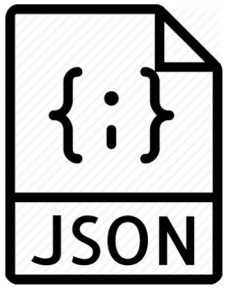
# Check Status


```
{
 "OK": true,
  "Errors": [],
  "CheckDetails": {
    "dns-status-check-external": {
      "OK": true,
      "Errors": [],
      "Namespace": "kuberhealthy",
      "LastRun": "2019-10-30T17:45:02.199703616Z",
      "AuthoritativePod": "kuberhealthy-5bfbb7f8b-6hkgs"
    },
```


Check ✅ or ❌ Bool

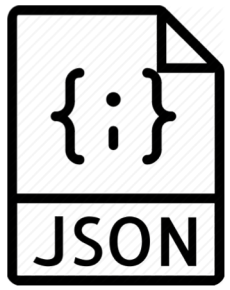
List of Errors


```
    "pod-restarts-check": {
      "OK": true,
      "Errors": [],
      "Namespace": "kuberhealthy",
      "LastRun": "2019-10-30T17:44:56.718123421Z",
      "AuthoritativePod": "kuberhealthy-5bfbb7f8b-t9kfs"
    }
  },
  "CurrentMaster": "kuberhealthy-5bfbb7f8b-6hkgs"
}
```

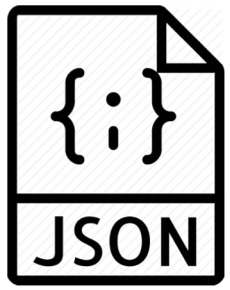
# Check Status

Check ✅ or ❌ Bool




# Check Status

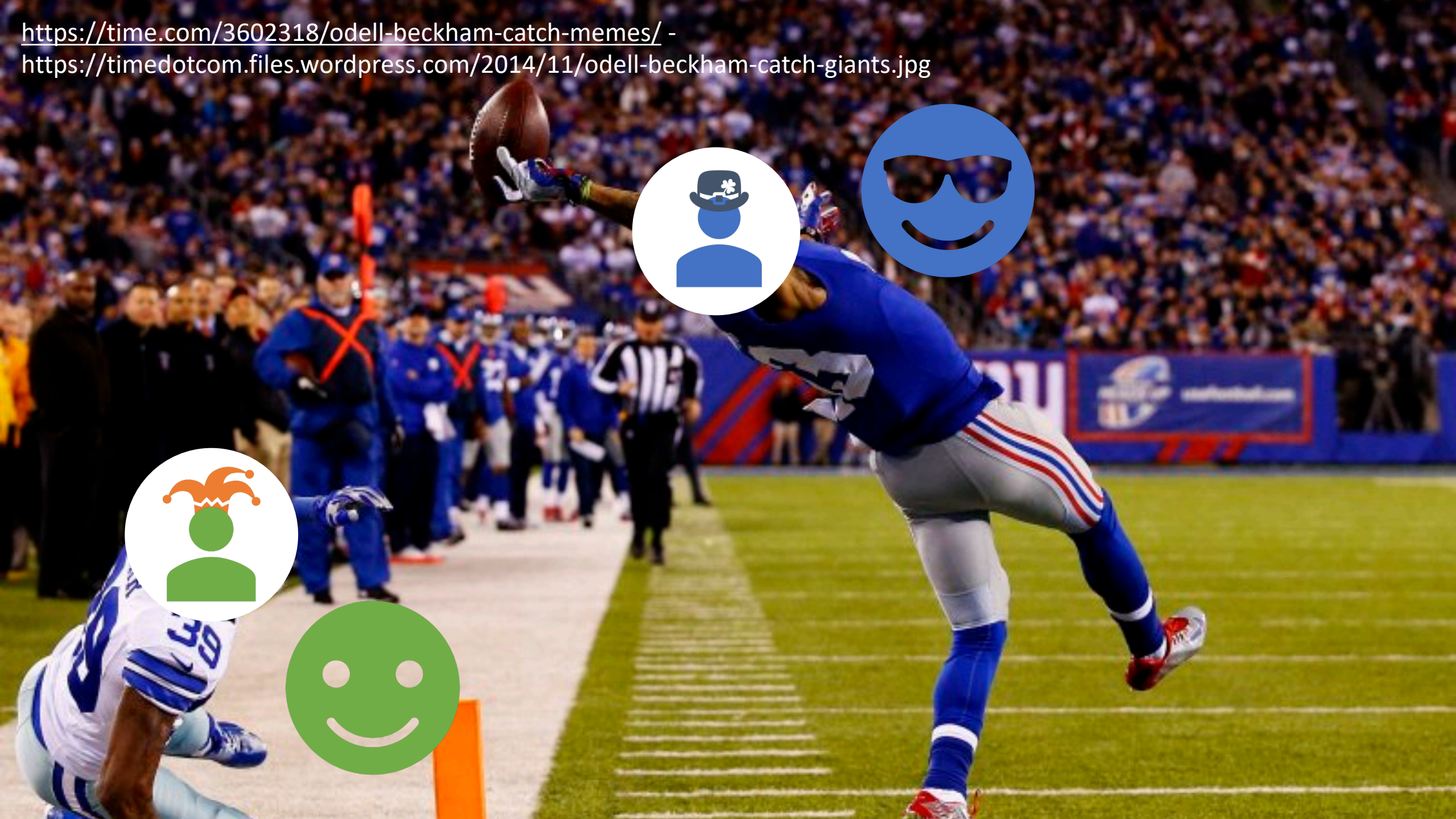
Check ✅ or ❌ Bool

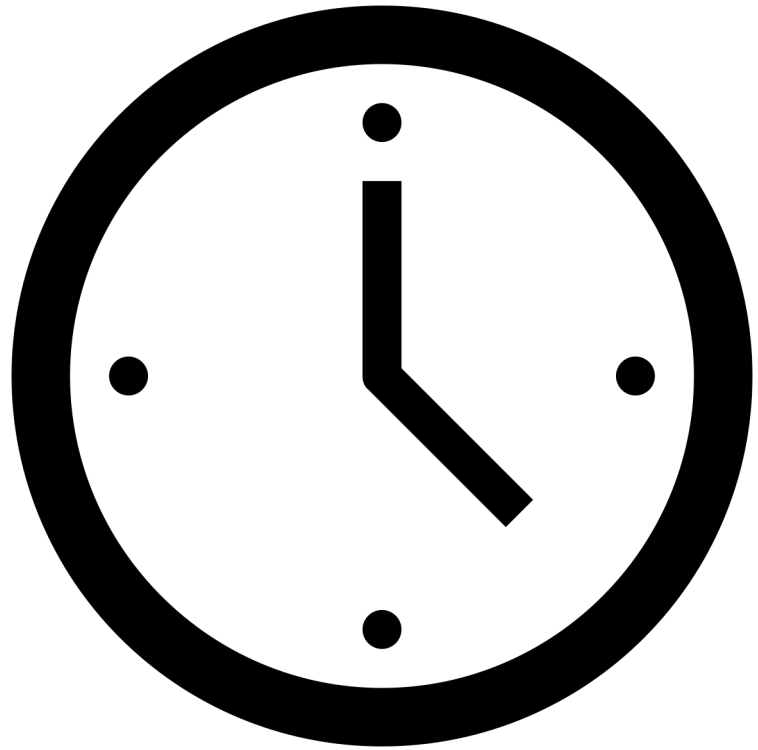


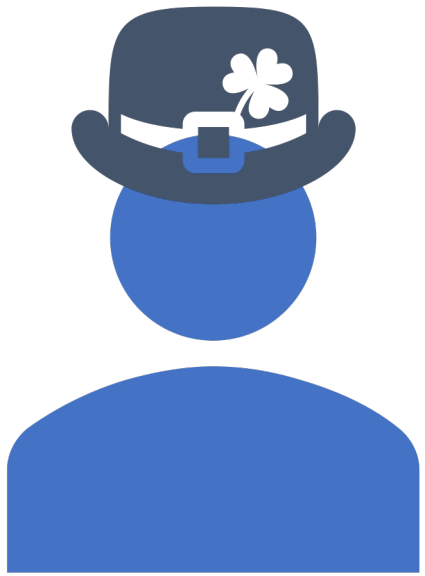


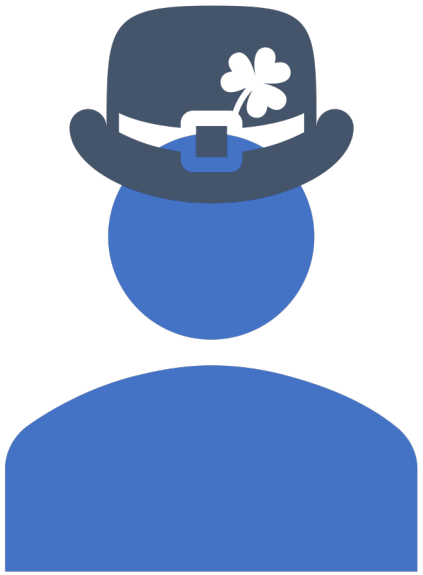
kuberhealthy

https://time.com/3602318/odell-beckham-catch-memes/ -
https://timedotcom.files.wordpress.com/2014/11/odell-beckham-catch-giants.jpg

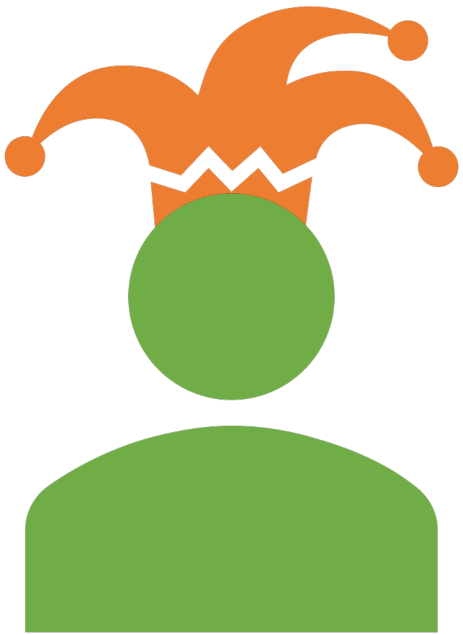






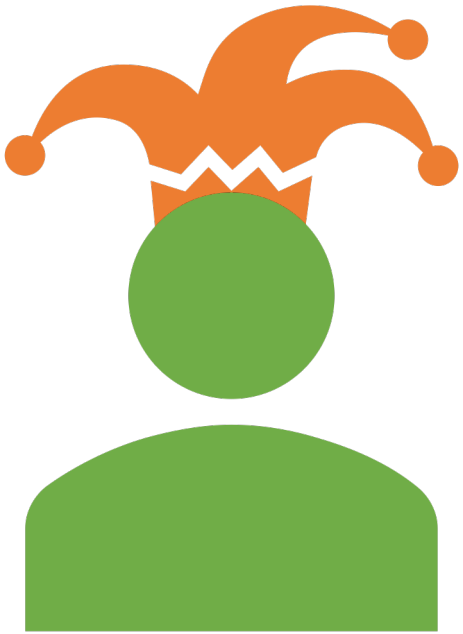


kuberhealthy



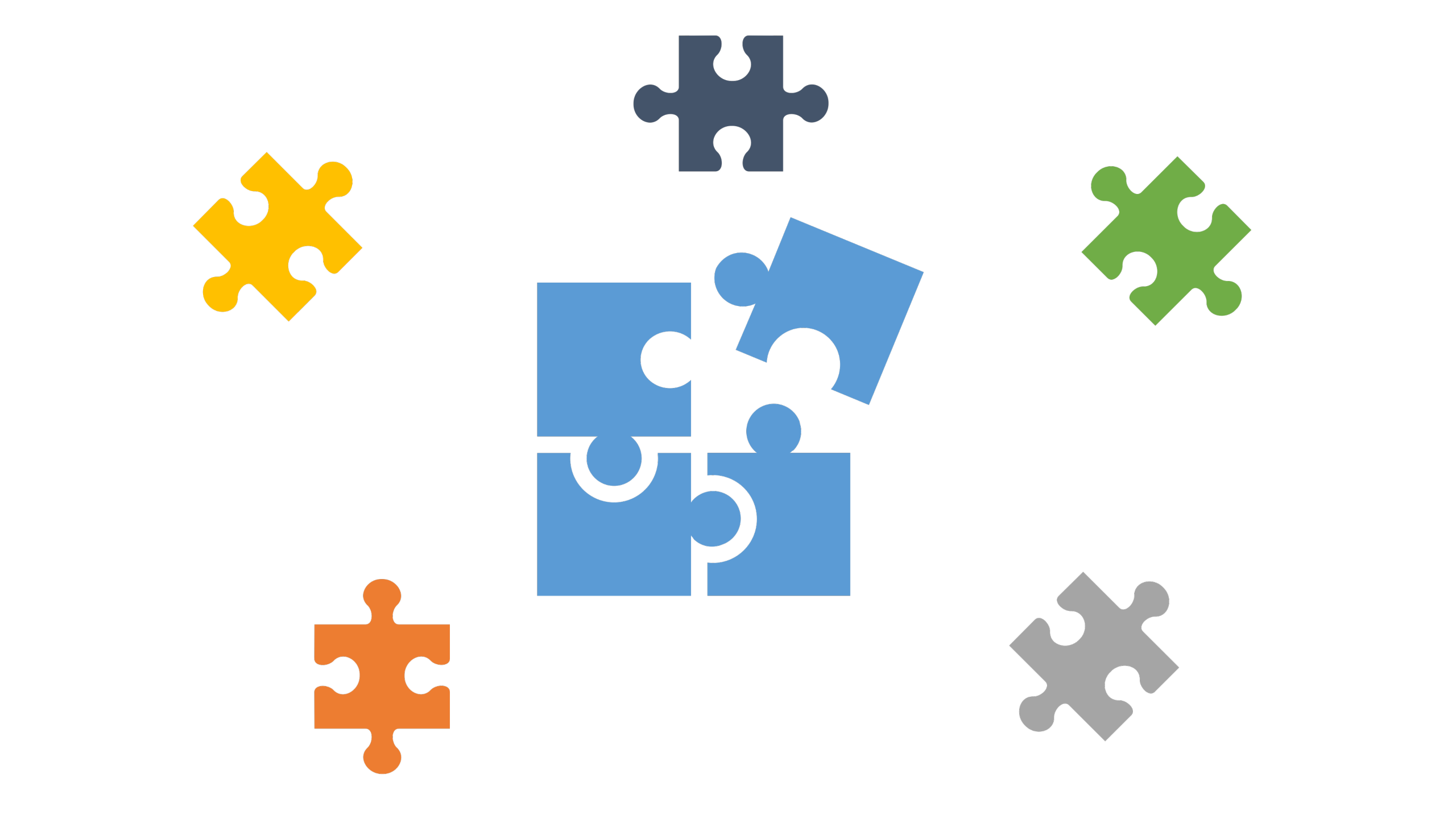
kuberhealthy

SYNTHETIC MONITOR?



# kuberhealthy

# 2.0.0



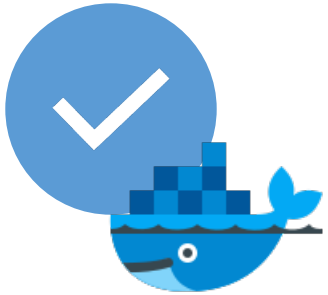

REMOVE
ALL THE CHECKS
imgflip.com

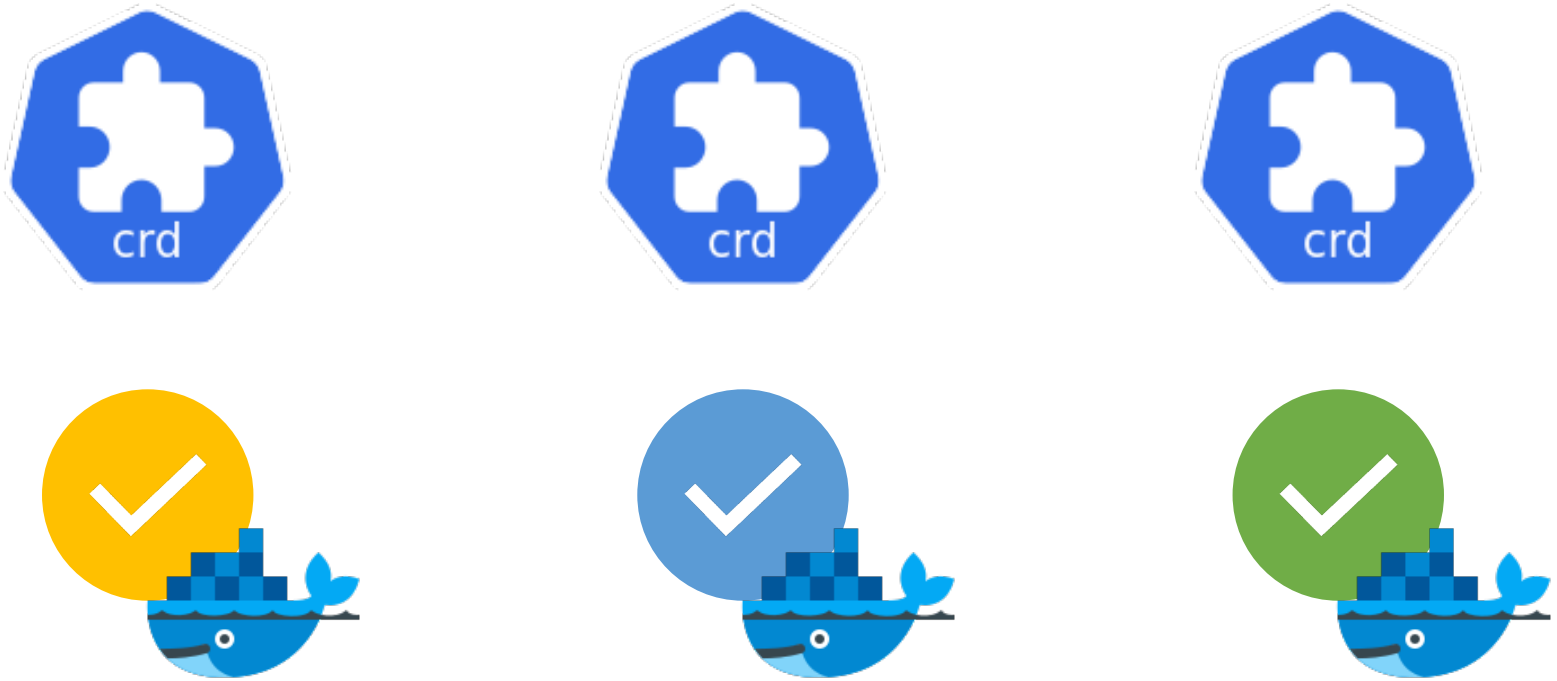
Create the ability to add
**external checks**


# KuberhealthyCheck - KHCheck


crd


crd




crd
crd
crd

# The KHCheck Anatomy

```yaml
apiVersion: comcast.github.io/v1
kind: KuberhealthyCheck
metadata:
  name: deployment
  namespace: kuberhealthy
spec:
  runInterval: 5m
  timeout: 10m
  extraAnnotations:
    comcast.com/testAnnotation: test.annotation
  extraLabels:
    testLabel: testLabel
  podSpec:
    containers:
    - name: deployment
      image: quay.io/comcast/deployment-check:1.0.0
      imagePullPolicy: IfNotPresent
      env:
        - name: CHECK_DEPLOYMENT_REPLICAS
          value: "4"
        - name: CHECK_DEPLOYMENT_ROLLING_UPDATE
          value: "false"
      resources:
        requests:
          cpu: 25m
          memory: 15Mi
        limits:
          cpu: 40m
      restartPolicy: Always
    serviceAccountName: deployment-khcheck
    terminationGracePeriodSeconds: 120
```

# Resource

```yaml
apiVersion: comcast.github.io/v1
kind: KuberhealthyCheck
```

# Metadata

```yaml
apiVersion: comcast.github.io/v1
kind: KuberhealthyCheck
metadata:
  name: deployment
  namespace: kuberhealthy
```

# Check name

```yaml
apiVersion: comcast.github.io/v1
kind: KuberhealthyCheck
metadata:
  name: deployment
  namespace: kuberhealthy
```

# Check namespace

```yaml
apiVersion: comcast.github.io/v1
kind: KuberhealthyCheck
metadata:
  name: deployment
  namespace: kuberhealthy
```

# KHCheck Spec

```yaml
apiVersion: comcast.github.io/v1
kind: KuberhealthyCheck
metadata:
  name: deployment
  namespace: kuberhealthy
spec:
  runInterval: 5m
  timeout: 10m
```

# Run Interval

```yaml
apiVersion: comcast.github.io/v1
kind: KuberhealthyCheck
metadata:
  name: deployment
  namespace: kuberhealthy
spec:
  runInterval: 5m
  timeout: 10m
```

# Timeout

```yaml
apiVersion: comcast.github.io/v1
kind: KuberhealthyCheck
metadata:
  name: deployment
  namespace: kuberhealthy
spec:
  runInterval: 5m
  timeout: 10m
```

# Extra Annotations and Labels

```yaml
apiVersion: comcast.github.io/v1
kind: KuberhealthyCheck
metadata:
  name: deployment
  namespace: kuberhealthy
spec:
  runInterval: 5m
  timeout: 10m
  extraAnnotations:
    comcast.com/testAnnotation: test.annotation
  extraLabels:
    testLabel: testLabel
```

# Pod Spec

```yaml
apiVersion: comcast.github.io/v1
kind: KuberhealthyCheck
metadata:
  name: deployment
  namespace: kuberhealthy
spec:
  runInterval: 5m
  timeout: 10m
  extraAnnotations:
    comcast.com/testAnnotation: test.annotation
  extraLabels:
    testLabel: testLabel
  podSpec:
    containers:
    - name: deployment
      image: quay.io/comcast/deployment-check:1.0.0
      imagePullPolicy: IfNotPresent
      env:
        - name: CHECK_DEPLOYMENT_REPLICAS
          value: "4"
        - name: CHECK_DEPLOYMENT_ROLLING_UPDATE
          value: "false"
      resources:
        requests:
          cpu: 25m
          memory: 15Mi
        limits:
          cpu: 40m
      restartPolicy: Always
    serviceAccountName: deployment-khcheck
    terminationGracePeriodSeconds: 120
```

```yaml
apiVersion: comcast.github.io/v1
kind: KuberhealthyCheck
metadata:
  name: deployment
  namespace: kuberhealthy

  podSpec:
    containers:
    - name: deployment
      image: quay.io/comcast/deployment-check:1.0.0
      imagePullPolicy: IfNotPresent
      env:
        - name: CHECK_DEPLOYMENT_REPLICAS
          value: "4"
        - name: CHECK_DEPLOYMENT_ROLLING_UPDATE
          value: "false"
      resources:
        requests:
          cpu: 25m
          memory: 15Mi
        limits:
          cpu: 40m
      restartPolicy: Always
    serviceAccountName: deployment-khcheck
    terminationGracePeriodSeconds: 120
```

# Container

```yaml
apiVersion: comcast.github.io/v1
kind: KuberhealthyCheck
metadata:
  name: deployment
  namespace: kuberhealthy

  podSpec:
    containers:
    - name: deployment
      image: quay.io/comcast/deployment-check:1.0.0
      imagePullPolicy: IfNotPresent
      env:
        - name: CHECK_DEPLOYMENT_REPLICAS
          value: "4"
        - name: CHECK_DEPLOYMENT_ROLLING_UPDATE
          value: "false"
      resources:
        requests:
          cpu: 25m
          memory: 15Mi
        limits:
          cpu: 40m
      restartPolicy: Always
    serviceAccountName: deployment-khcheck
    terminationGracePeriodSeconds: 120
```

# Check image

```yaml
apiVersion: comcast.github.io/v1
kind: KuberhealthyCheck
metadata:
  name: deployment
  namespace: kuberhealthy

  podSpec:
    containers:
    - name: deployment
      image: quay.io/comcast/deployment-check:1.0.0
      imagePullPolicy: IfNotPresent
      env:
        - name: CHECK_DEPLOYMENT_REPLICAS
          value: "4"
        - name: CHECK_DEPLOYMENT_ROLLING_UPDATE
          value: "false"
      resources:
        requests:
          cpu: 25m
          memory: 15Mi
        limits:
          cpu: 40m
      restartPolicy: Always
    serviceAccountName: deployment-khcheck
    terminationGracePeriodSeconds: 120
```

# The KHCheck Anatomy

```yaml
apiVersion: comcast.github.io/v1
kind: KuberhealthyCheck
metadata:
  name: deployment
  namespace: kuberhealthy
spec:
  runInterval: 5m
  timeout: 10m
  extraAnnotations:
    comcast.com/testAnnotation: test.annotation
  extraLabels:
    testLabel: testLabel
  podSpec:
    containers:
    - name: deployment
      image: quay.io/comcast/deployment-check:1.0.0
      imagePullPolicy: IfNotPresent
      env:
        - name: CHECK_DEPLOYMENT_REPLICAS
          value: "4"
        - name: CHECK_DEPLOYMENT_ROLLING_UPDATE
          value: "false"
      resources:
        requests:
          cpu: 25m
          memory: 15Mi
        limits:
          cpu: 40m
      restartPolicy: Always
    serviceAccountName: deployment-khcheck
    terminationGracePeriodSeconds: 120
```


# Check Pod

KH_REPORTING_URL

# KH_REPORTING_URL

http://kuberhealthy.kuberhealthy.svc.cluster.local/externalCheckStatus

POST

KH_REPORTING_URL

POST

KH_REPORTING_URL


```
{
  "OK": false,
  "Errors": [
    "Error 1 here",
    "Error 2 here"
  ]
}
```

```go
import (
    checkclient "github.com/Comcast/kuberhealthy/pkg/checks/external/checkClient"
)

// runChecks sleeps given the duration set in the environment variable REPORT_DELAY and
// returns a bool set by the environment variable REPORT_FAILURE
func runChecks() bool {
    time.Sleep(reportDelay)
    return reportFailure
}

func main() {

    log.Println("Using kuberhealthy reporting url", os.Getenv(external.KHReportingURL))
    log.Println("Waiting", reportDelay, "seconds before reporting...")

    reportFailure = runChecks()

    var err error
    if reportFailure {
        log.Println("Reporting failure...")
        err = checkclient.ReportFailure([]string{"Test has failed!"})
    } else {
        log.Println("Reporting success...")
        err = checkclient.ReportSuccess()
    }

    if err != nil {
        log.Println("Error reporting to Kuberhealthy servers:", err)
        return
    }
    log.Println("Successfully reported to Kuberhealthy servers")
}
```

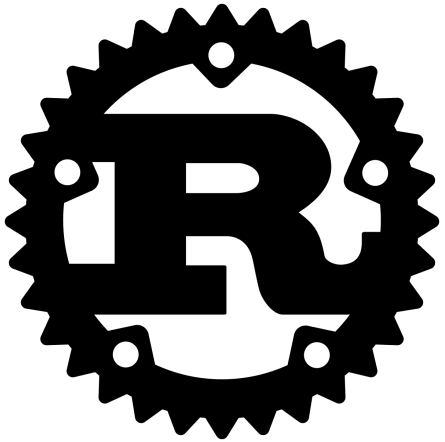
```
import (
   checkclient "github.com/Comcast/kuberhealthy/pkg/checks/external/checkClient"
)

// runChecks sleeps given the duration set in the environment variable REPORT_DELAY and
// returns a bool set by the environment variable REPORT_FAILURE
func runChecks() bool {
   time.Sleep(reportDelay)
   return reportFailure
}

func main() {

   log.Println("Using kuberhealthy reporting url", os.Getenv(external.KHReportingURL))
   log.Println("Waiting", reportDelay, "seconds before reporting...")

   reportFailure = runChecks()

   var err error
   if reportFailure {
      log.Println("Reporting failure...")
      err = checkclient.ReportFailure([]string{"Test has failed!"})
   } else {
      log.Println("Reporting success...")
      err = checkclient.ReportSuccess()
   }

   if err != nil {
      log.Println("Error reporting to Kuberhealthy servers:", err)
      return
   }
   log.Println("Successfully reported to Kuberhealthy servers")
}
```

```go
import (
    checkclient "github.com/Comcast/kuberhealthy/pkg/checks/external/checkClient"
)

// runChecks sleeps given the duration set in the environment variable REPORT_DELAY and
// returns a bool set by the environment variable REPORT_FAILURE
func runChecks() bool {
    time.Sleep(reportDelay)
    return reportFailure
}

func main() {

    log.Println("Using kuberhealthy reporting url", os.Getenv(external.KHReportingURL))
    log.Println("Waiting", reportDelay, "seconds before reporting...")

    reportFailure = runChecks()

    var err error
    if reportFailure {
        log.Println("Reporting failure...")
        err = checkclient.ReportFailure([]string{"Test has failed!"})
    } else {
        log.Println("Reporting success...")
        err = checkclient.ReportSuccess()
    }

    if err != nil {
        log.Println("Error reporting to Kuberhealthy servers:", err)
        return
    }
    log.Println("Successfully reported to Kuberhealthy servers")
}
```

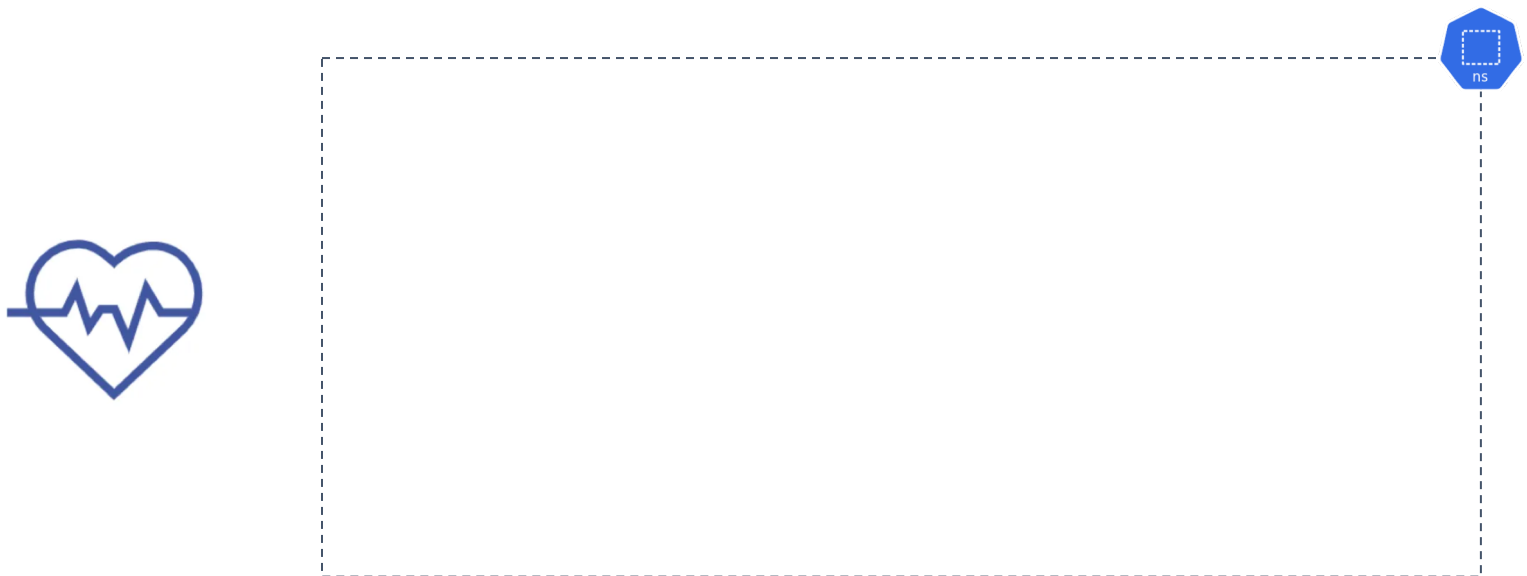
```go
import (
    checkclient "github.com/Comcast/kuberhealthy/pkg/checks/external/checkClient"
)

// runChecks sleeps given the duration set in the environment variable REPORT_DELAY and
// returns a bool set by the environment variable REPORT_FAILURE
func runChecks() bool {
    time.Sleep(reportDelay)
    return reportFailure
}

func main() {

    log.Println("Using kuberhealthy reporting url", os.Getenv(external.KHReportingURL))
    log.Println("Waiting", reportDelay, "seconds before reporting...")

    reportFailure = runChecks()

    var err error
    if reportFailure {
        log.Println("Reporting failure...")
        err = checkclient.ReportFailure([]string{"Test has failed!"})
    } else {
        log.Println("Reporting success...")
        err = checkclient.ReportSuccess()
    }

    if err != nil {
        log.Println("Error reporting to Kuberhealthy servers:", err)
        return
    }
    log.Println("Successfully reported to Kuberhealthy servers")
}
```

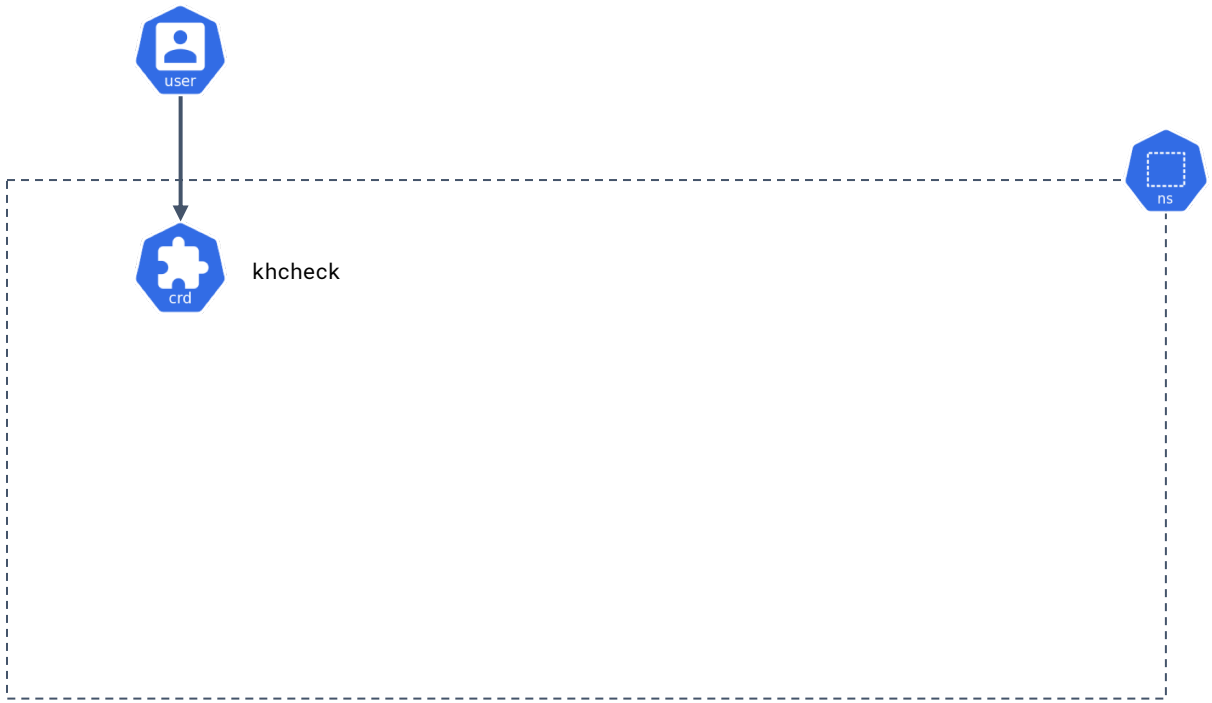


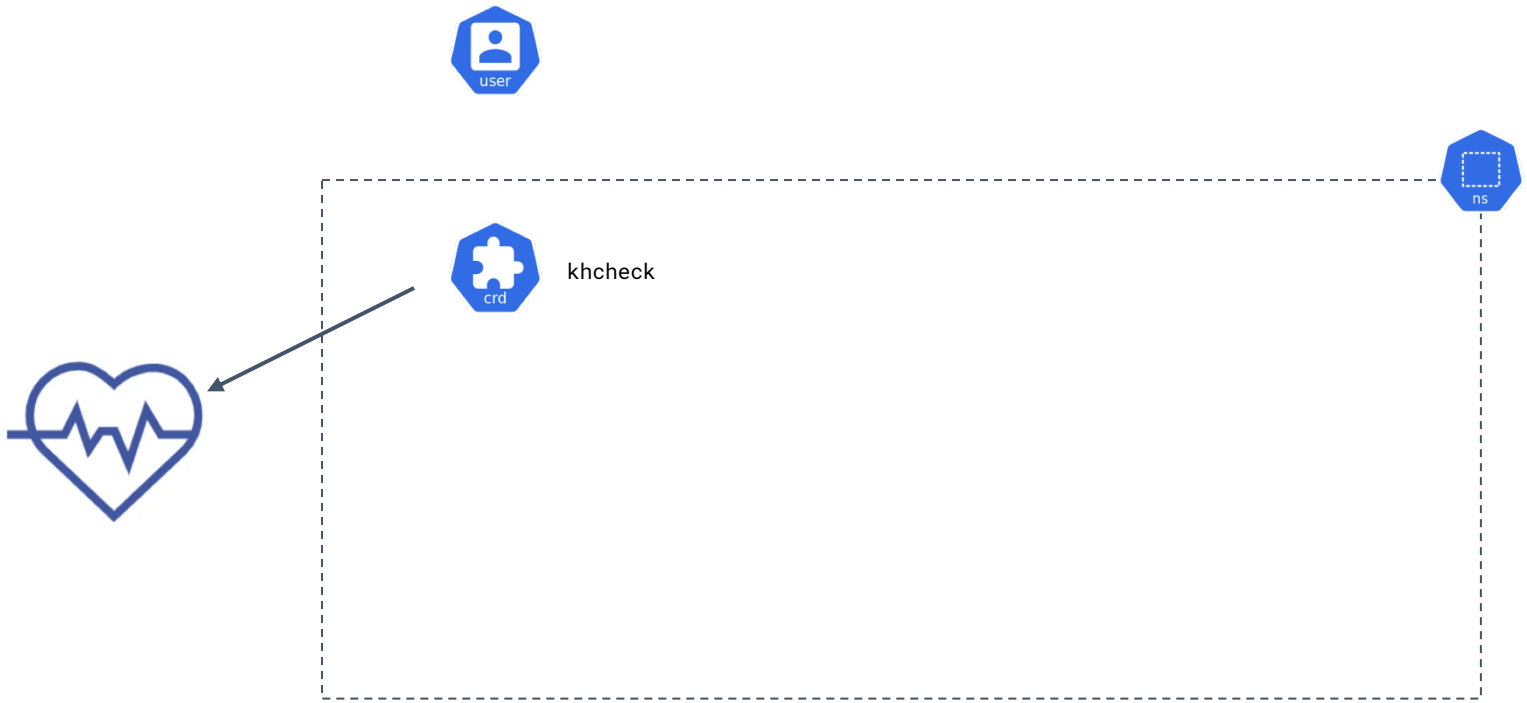
**PSA:** More client libraries in different languages to report KH checks would be appreciated!

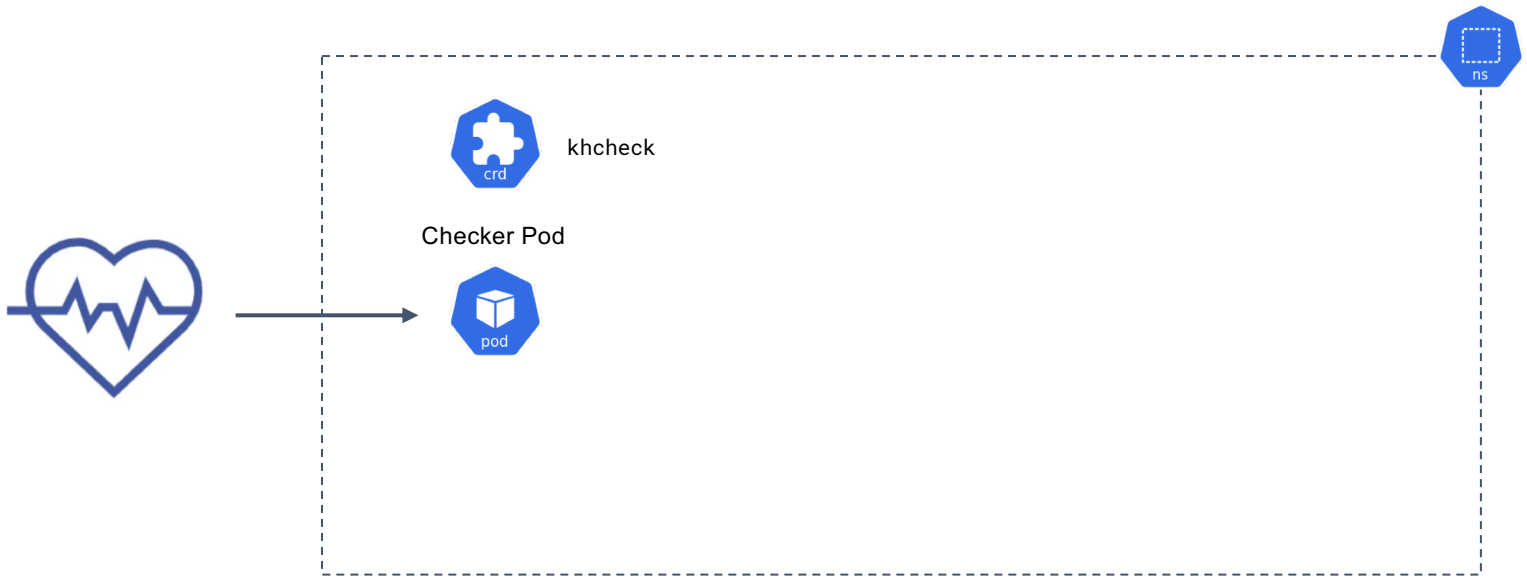


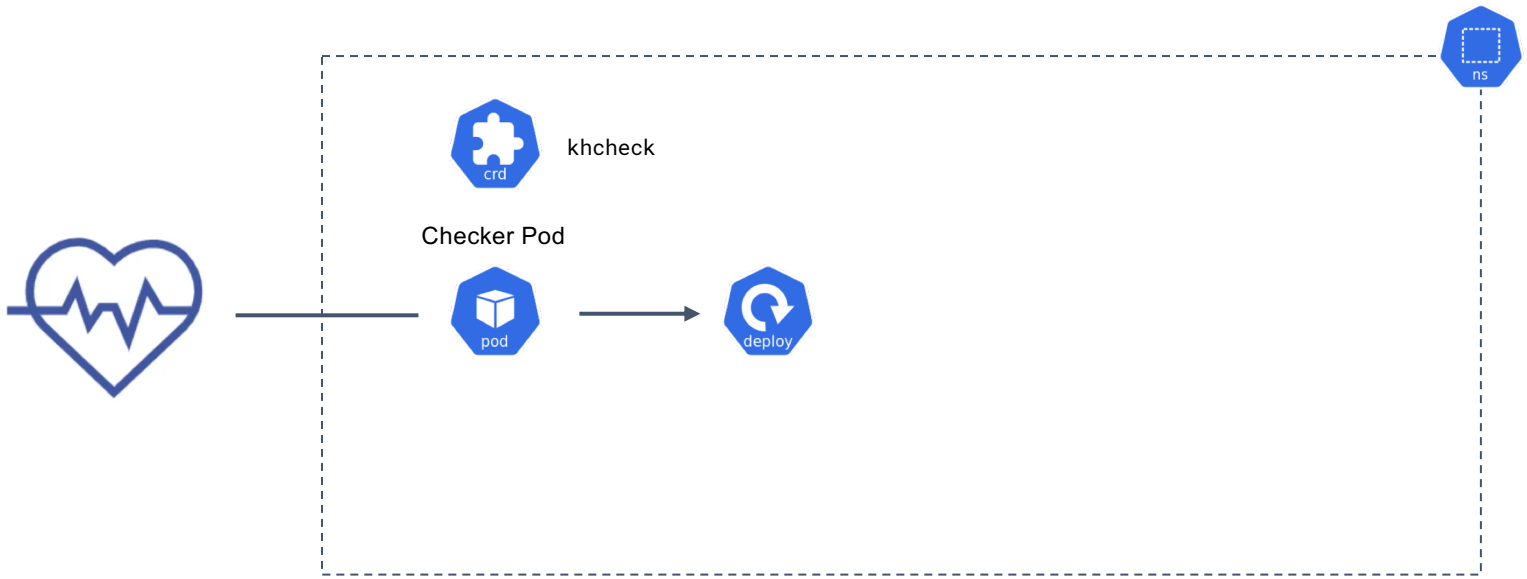

# Deployment Check

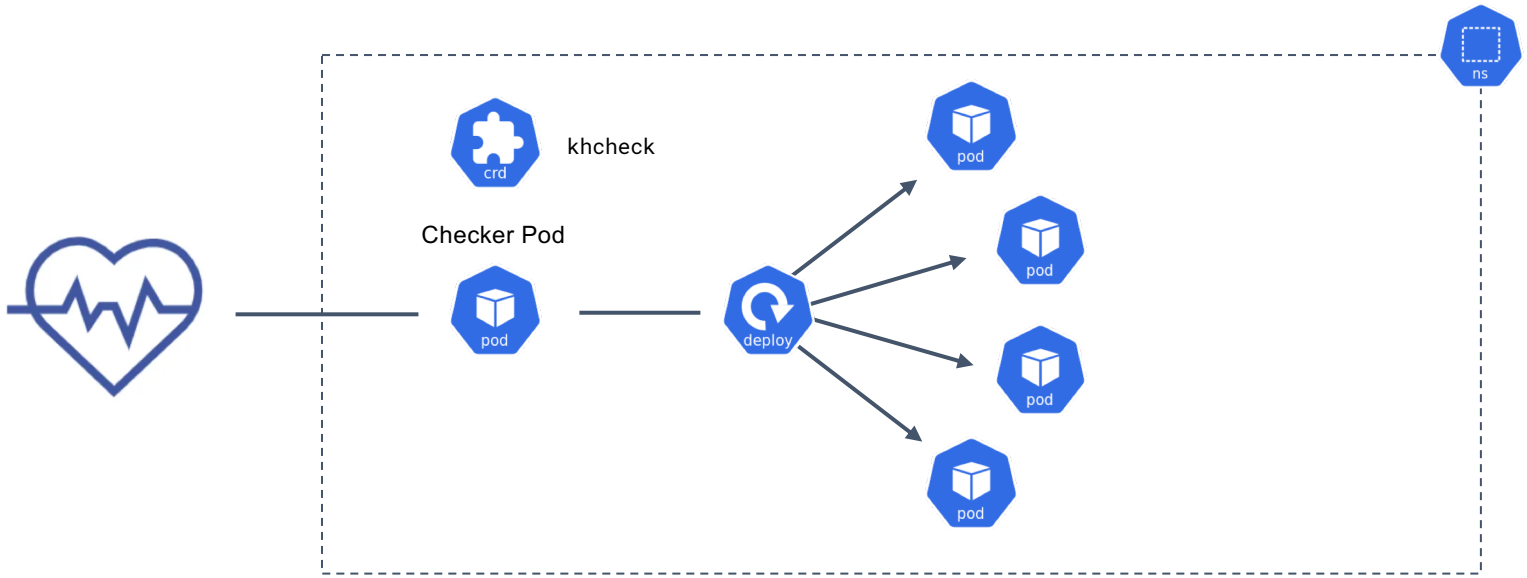

ns

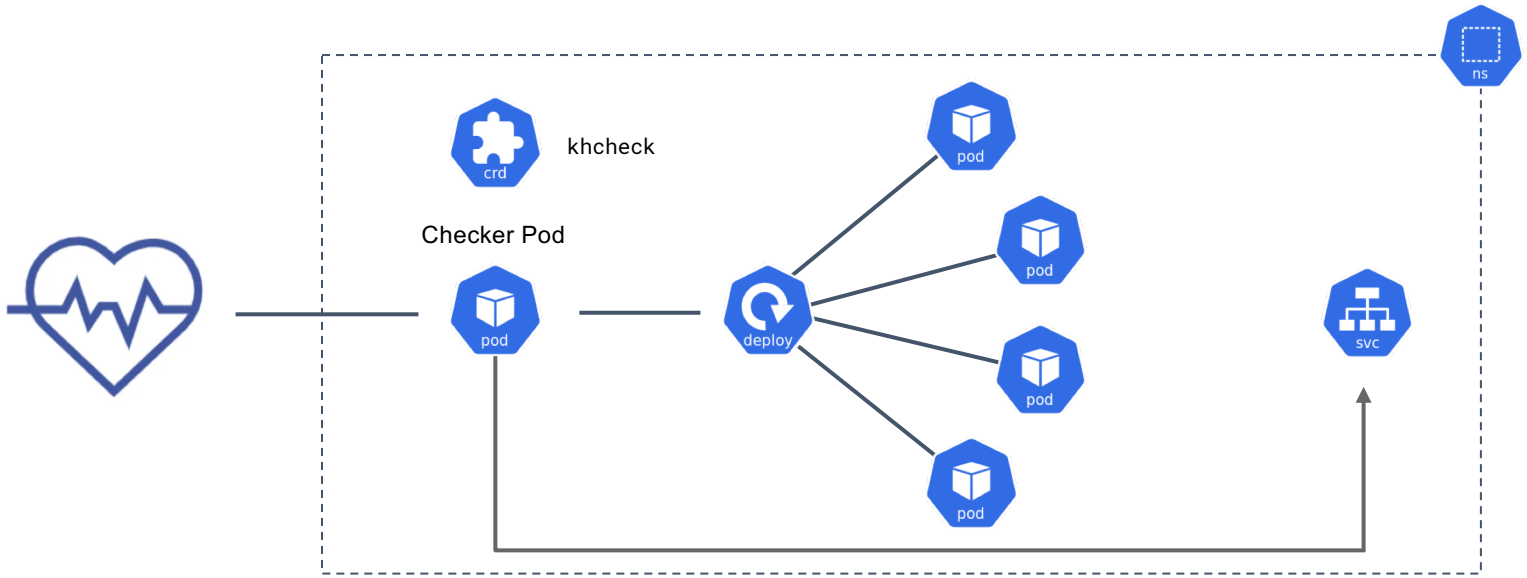

user
ns
crd
khcheck

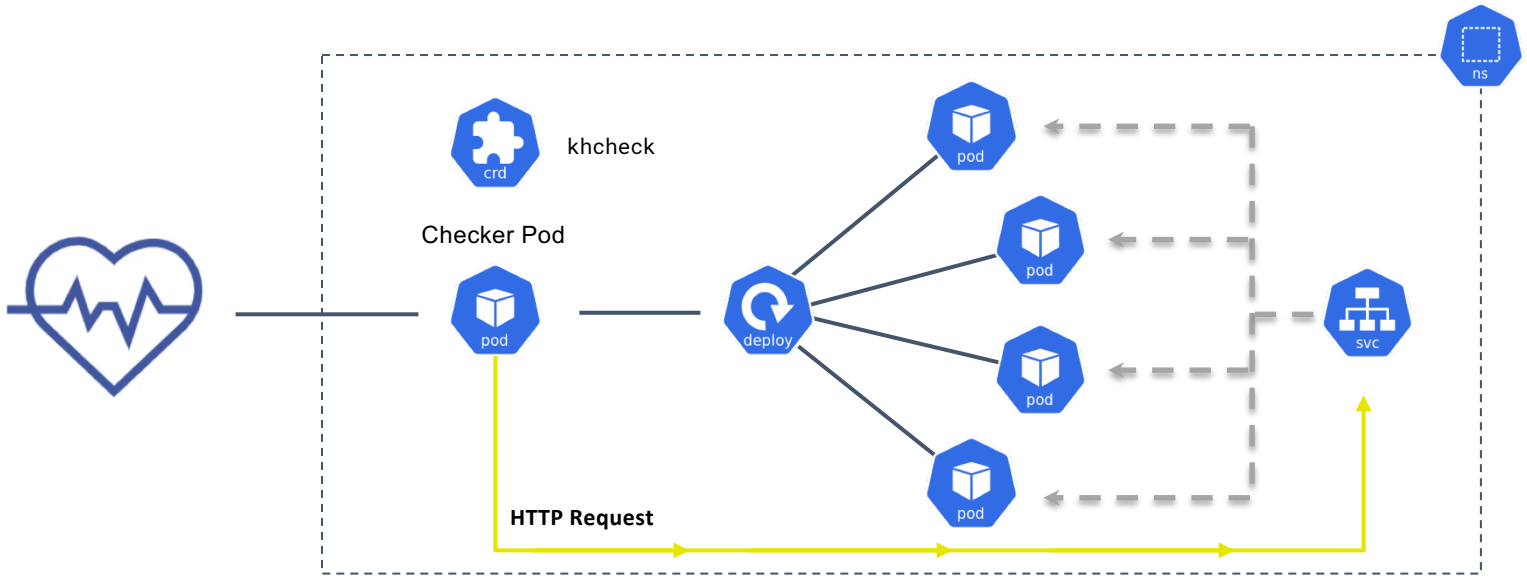

user
ns
crd
khcheck

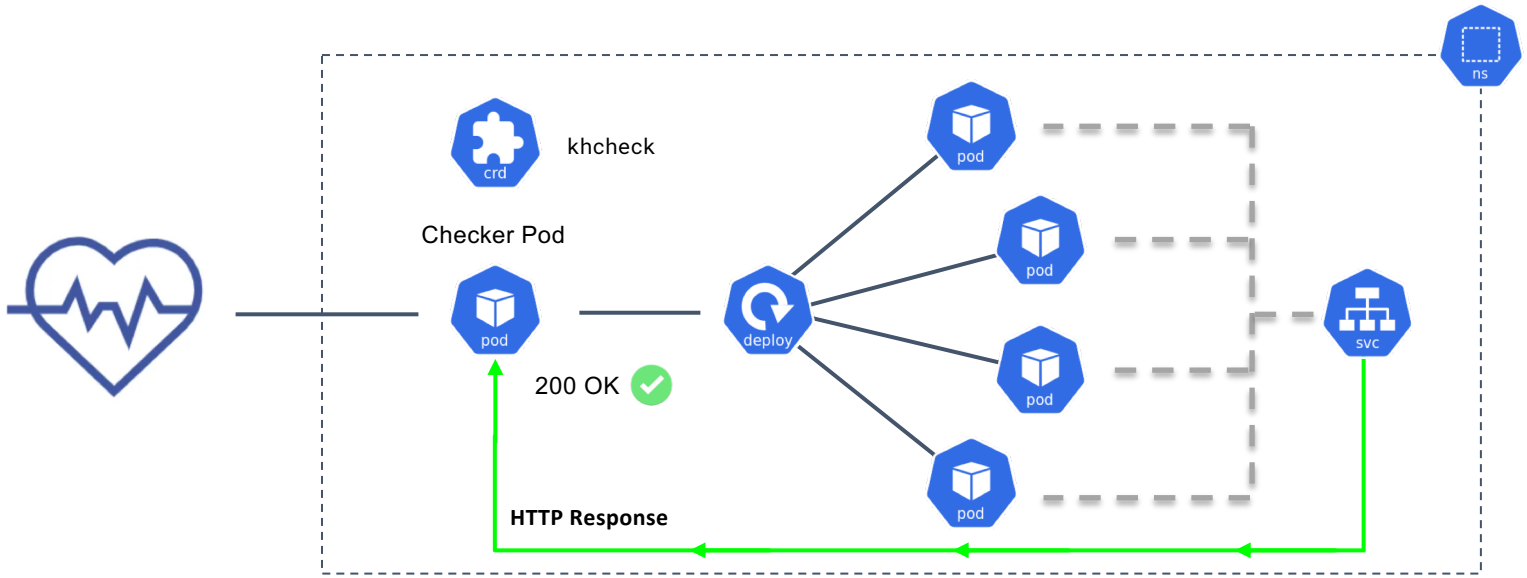

ns
crd
khcheck
Checker Pod
pod

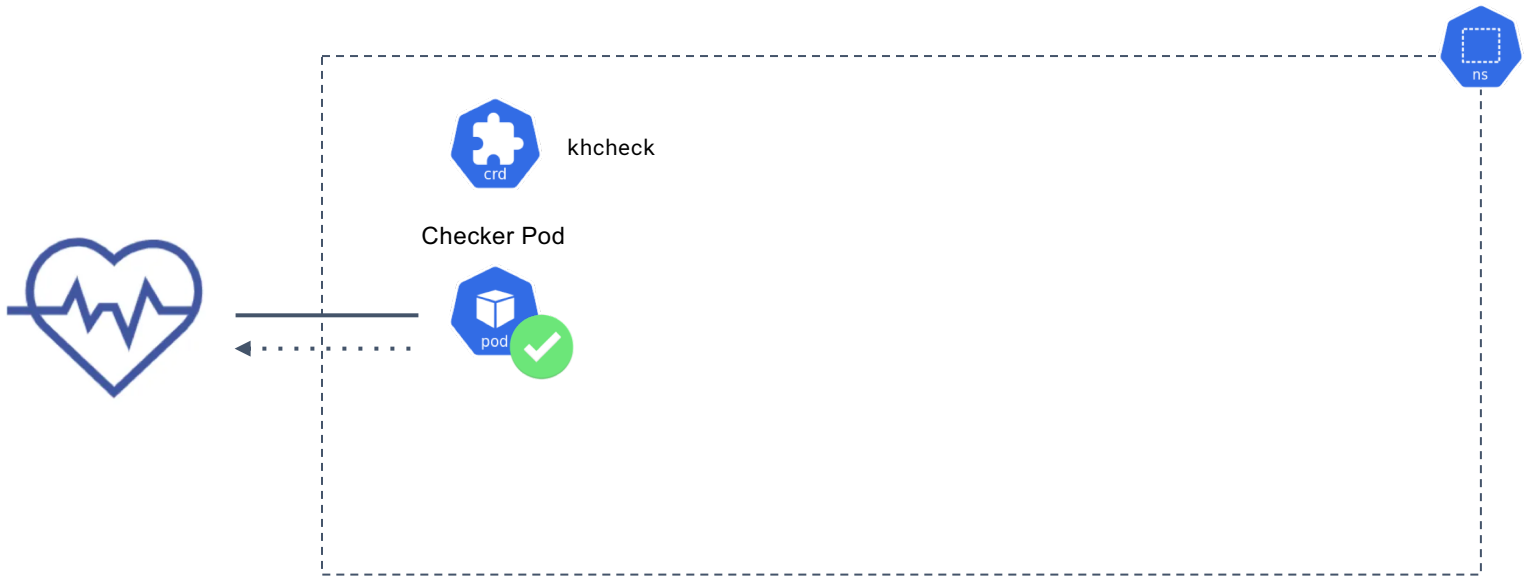

ns
crd
khcheck
Checker Pod
pod
deploy

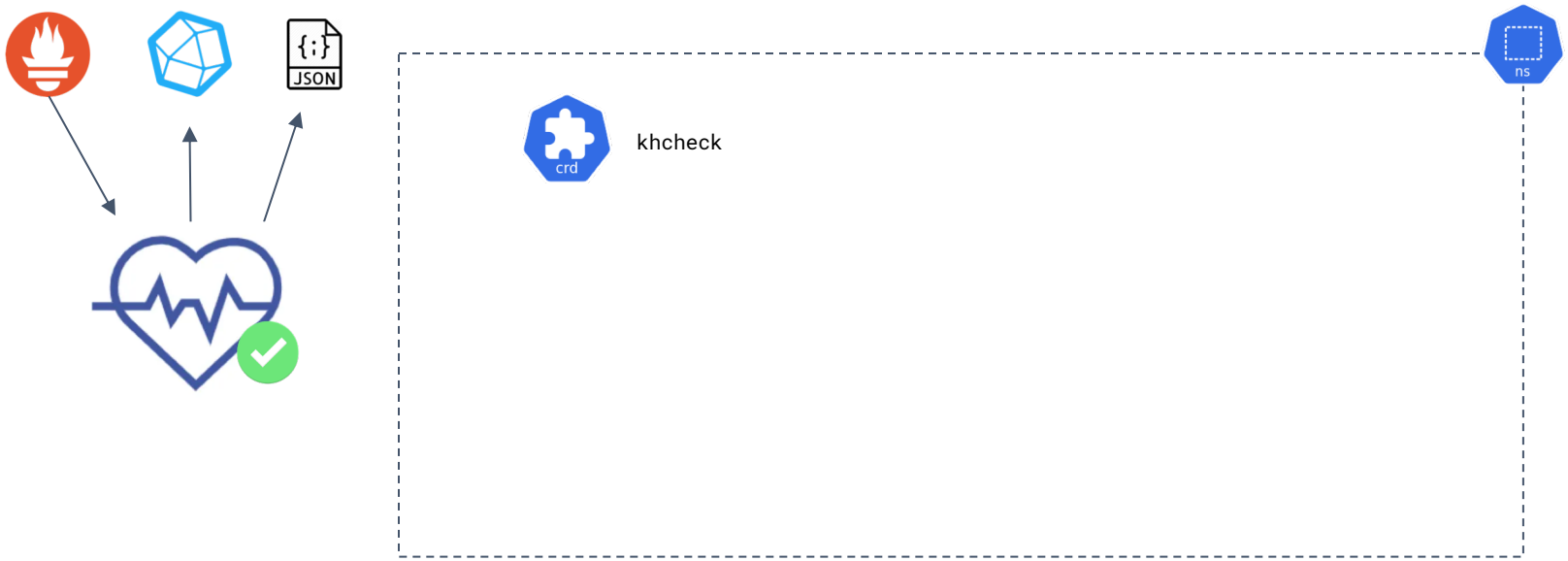

ns
crd
khcheck
Checker Pod
pod
deploy
pod
pod
pod
pod


ns
crd
khcheck
Checker Pod
pod
pod
pod
pod
pod
deploy
svc


ns
crd
khcheck
Checker Pod
pod
deploy
pod
pod
pod
pod
svc
HTTP Request


ns
crd
khcheck
Checker Pod
pod
deploy
pod
pod
pod
svc
200 OK
pod
HTTP Response


ns
crd
khcheck
Checker Pod
pod


JSON
ns
crd
khcheck

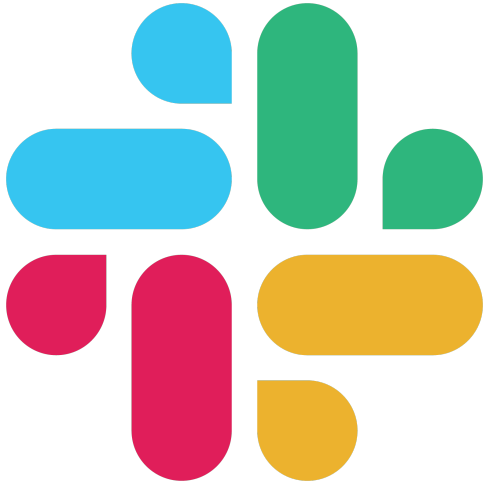
## External Checks Registry

Here is a list of external checks you can apply to your kubernetes cluster once you have Kuberhealthy installed. For convenient addition directly from the web, ensure you have Kuberhealthy in your cluster and run `kubectl apply -f` on the `khcheck` resource URL. For easy cleanup, just run `kubectl delete -f` on the `khcheck` resource URL.

Make sure to add your check here:

| Check Name | Description | khcheck Resource | Contributor |
|---|---|---|---|
| Daemonset Check | Ensures daemonsets can be successfully deployed | daemonSetCheck.yaml | @integrii @joshulyne |
| Deployment Test | Ensures that a Deployment and Service can be provisioned, created, and serve traffic within the Kubernetes cluster | deployment-check.yaml | @jonnydawg |
| Pod Restarts Check | Checks for excessive pod restarts in any namespace | podRestartsCheck.yaml | @integrii @joshulyne |
| Pod Status Check | Checks for unhealthy pod statuses in a target namespace | podStatusCheck.yaml | @integrii @rukatm |

Comcast / kuberhealthy



# kuberhealthy

Thank you!