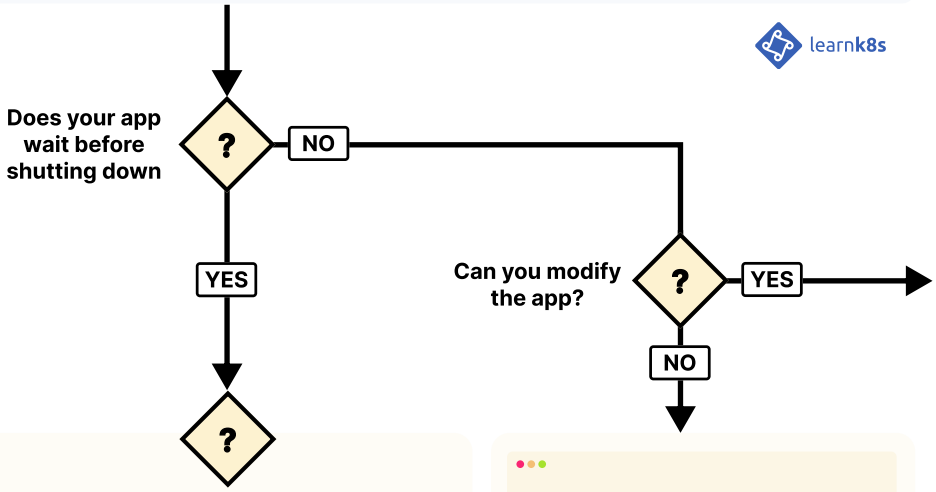
I SEE 5XX ERRORS IN MY KUBERNETES DEPLOYMENTS

This flowchart helps you debug issues with pod graceful shutdown in Kubernetes.



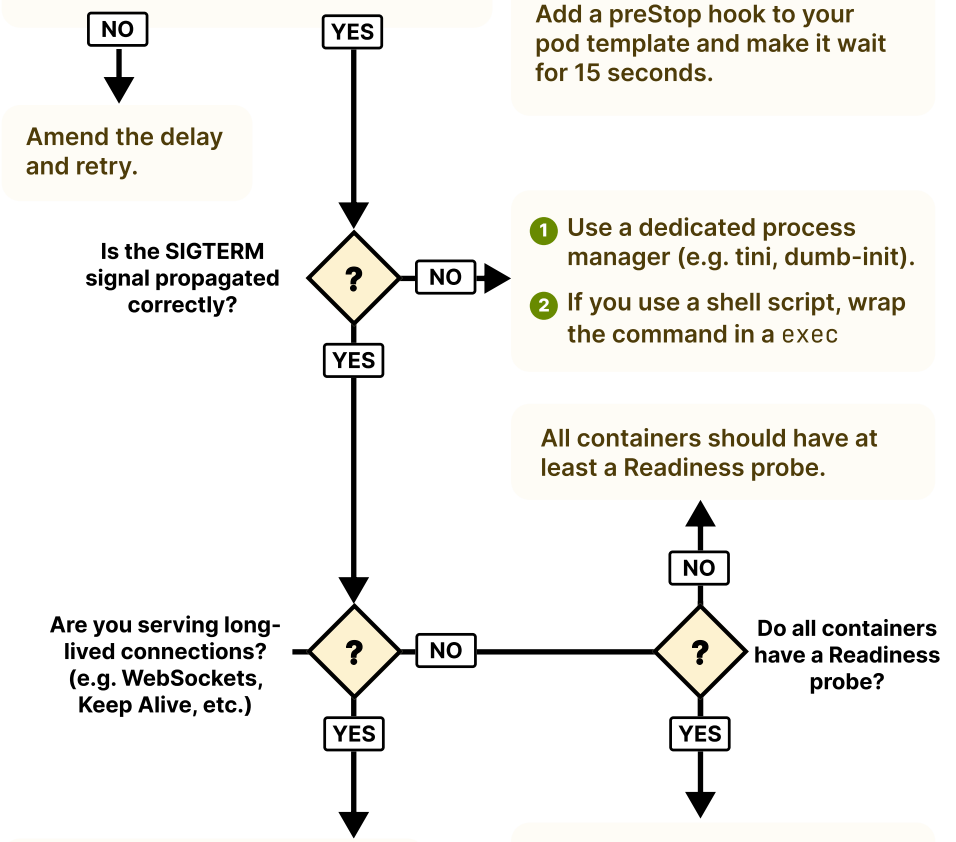
1. The apps waits for 15 seconds before shutting down.
2. The app delay is strictly less than the `terminationGracePeriodSeconds`

OR

2. preHook delay + 15 seconds is strictly less than the `terminationGracePeriodSeconds`

```
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
    - name: web
      image: nginx
      ports:
        - name: web
          containerPort: 80
      lifecycle:
        preStop:
          exec:
            command: ["sleep", "15"]
```

Add a preStop hook to your pod template and make it wait for 15 seconds.



🌈 Rainbow deployments

You should consider using rainbow deployments instead of the regular rolling updates in Kubernetes.

More info is needed

There is a component in the cluster using the endpoint despite it being removed. This may require further investigation and may be depended on your setup.

You should wait for 15 seconds and, in the next 15 seconds, close any connections and shut down the application.

Go

```
package main

import (
    "fmt"
    "os"
    "os/signal"
    "syscall"
)

func main() {

    sigs := make(chan os.Signal, 1)
    done := make(chan bool, 1)
    //registers the channel
    signal.Notify(sigs, syscall.SIGTERM)

    go func() {
        sig := <-sigs
        fmt.Println("Caught SIGTERM, shutting down")
        // Finish any outstanding requests, then...
        done <- true
    }()

    fmt.Println("Starting application")
    // Main logic goes here
    <-done
    fmt.Println("exiting")
}
```

Python

```
import signal, time, os

def shutdown(signum, frame):
    print('Caught SIGTERM, shutting down')
    # Finish any outstanding requests, then...
    exit(0)

if __name__ == '__main__':
    # Register handler
    signal.signal(signal.SIGTERM, shutdown)
    # Main logic goes here
```

Node.js

```
process.on('SIGTERM', () => {
  console.log('The service is about to shut down!');

  // Finish any outstanding requests, then...
  process.exit(0);
});
```

Java

```
public class App {

  public static void main(String[] args) {

    var shutdownListener = new Thread() {
      public void run() {
        // Main logic goes here
      }
    };
    Runtime.getRuntime().addShutdownHook(shutdownListener);
  }
}
```

.NET Core

```
internal class LifetimeEventsHostedService: IHostedService {
  private readonly IHostApplicationLifetime _appLifetime;

  public LifetimeEventsHostedService(
    ILogger < LifetimeEventsHostedService > logger,
    IHostApplicationLifetime appLifetime,
    TelemetryClient telemtryClient) {
    _appLifetime = appLifetime;
  }

  public Task StartAsync(CancellationToken cancellationToken) {
    _appLifetime.ApplicationStarted.Register(OnStarted);
    _appLifetime.ApplicationStopping.Register(OnStopping);
    _appLifetime.ApplicationStopped.Register(OnStopped);

    return Task.CompletedTask;
  }

  public Task StopAsync(CancellationToken cancellationToken) {
    return Task.CompletedTask;
  }

  private void OnStarted() {}

  private void OnStopping() {
    // Main logic here
  }

  private void OnStopped() {}
}
```