**HPA disruption workflow**  
1. cpu-stress   
2. Constent pod kill every 30 seconds   
3. create a workflow to execute these experiments simultaneously  
**What**  
To effectively disrupt and test the **Horizontal Pod Autoscaler (HPA)** behavior under stress conditions, we need to simulate real-world scenarios that could impact its scaling decisions. This workflow will combine **CPU stress testing** and **continuous pod termination** to evaluate how well HPA handles such disruptions.  
**Why**  
 **Evaluate HPA response to CPU stress**

* Verify if HPA correctly increases pod replicas under high CPU load.
* Ensure that scaling happens within expected thresholds.

**Assess system stability under frequent pod failures**

* Killing a pod every **30 seconds** simulates a real-world failure scenario.
* Verify if HPA scales up/down appropriately without causing service disruption.

**Check HPA behavior when both stressors are active**

* Determine if HPA can still maintain performance while handling multiple stressors simultaneously.

**yaml**

 apiVersion: chaos-mesh.org/v1alpha1  
kind: Workflow  
metadata:  
 name: try-workflow-parallel  
spec:  
 entry: kalyani  
 templates:  
    - name: kalyani  
      templateType: Parallel  
      deadline: 240s  
      children:  
        - workflow-stress-chaos  
        - workflow-pod-chaos-schedule  
    - name: workflow-pod-chaos-schedule  
      templateType: Schedule  
      deadline: 120s  
      schedule:  
        schedule: '@every 30s'  
        concurrencyPolicy: Forbid  
        type: 'PodChaos'  
        podChaos:  
          action: pod-kill  
          mode: all  
          selector:  
            labelSelectors:  
              'app': 'nginx'  
    - name: workflow-stress-chaos  
      templateType: StressChaos  
      deadline: 300s  
      stressChaos:  
        mode: all  
        selector:  
          labelSelectors:  
            'app': 'nginx'  
        stressors:  
          cpu:  
            workers: 1  
            load: 20  
            options: ['--cpu 1', '--timeout 600']

**How to Eradicate These Issues?  
  
1. Optimize HPA Configuration**

* Set appropriate CPU utilization targets (e.g., 50-70%) to ensure timely scaling.
* Tune minReplicas and maxReplicas based on load patterns.

**2. Implement Pod Disruption Budgets (PDB)**

* Prevent excessive pod terminations.
* Define a minimum available pod count.

**3. Use Readiness and Liveness Probes**

* Ensure Kubernetes waits before routing traffic to new pods.
* Restart unhealthy pods only when necessary.

**4. Optimize Scaling Policies**

* Use custom metrics via KEDA or Prometheus Adapter to improve autoscaling decisions.
* Monitor CPU, memory, request rate, and latency for better scaling signals.

**5. Monitor & Alert Using Observability Tools**

* Use Prometheus + Grafana for real-time metrics.
* Set up alerts for unusual scaling behavior.