**ResourceStarvation (Simulating Resource Starvation During Pod Shutdown)**  
CPU-Memory-Stress   
Slow-pod-kill.yaml   
Both of these experiments should run parallelly  
  
  
**What it means?**  
  
Running **Resource Starvation (CPU-Memory-Stress) and Slow Pod Kill experiments in parallel** means that we are simulating two failure scenarios at the same time:

1. **Resource Starvation (CPU-Memory-Stress)**
   * This experiment overloads the CPU and memory of a pod by consuming excessive resources, simulating high resource contention.
   * It helps evaluate how the application behaves under extreme resource exhaustion conditions.
2. **Slow Pod Kill**
   * This experiment delays the shutdown of a pod, which can impact the graceful termination process and cause cascading failures.
   * It helps identify issues with service availability, connection draining, and dependency handling when a pod takes too long to terminate.

**Why Do We Need This?**

Running both experiments **simultaneously** helps analyze:

* The **combined impact** of resource exhaustion and delayed termination on an application’s stability.
* How well the application **recovers** when faced with both **high resource load and slow shutdowns** at the same time.
* Whether the system has proper failover mechanisms like **Horizontal Pod Autoscaler (HPA), Pod Disruption Budgets (PDBs), and Graceful Termination** strategies.  
    
    
    
  **Path:** /root/Kalyani **cat ResourceStarvation.yaml**

**How to Eradicate This?**

To **mitigate and recover** from these issues, consider:

1. **For Resource Starvation (CPU-Memory-Stress):**
   * Implement **Resource Requests and Limits** in Kubernetes to prevent excessive resource usage.
   * Enable **Horizontal Pod Autoscaling (HPA)** to scale up resources during high demand.
   * Use **Node Autoscaler** to provision additional nodes if resource exhaustion happens frequently.
2. **For Slow Pod Kill:**
   * Optimize the **terminationGracePeriodSeconds** value for faster and controlled shutdown.
   * Implement **Readiness & Liveness Probes** to detect and recover from unhealthy states.
   * Use **Pod Disruption Budgets (PDBs)** to ensure minimal impact on the system during pod shutdowns.