Redis Deployment Patterns

Minimal Redis Deployment Patterns

Context: For use in early-stage, agent-driven DevOps workflows and LLM orchestration requiring lightweight Redis deployment on Kubernetes (using kind cluster).

Purpose of This Guide

This guide provides two minimal and valid Redis deployment examples for Kubernetes that can:

- Run on modest local machines (2 CPUs, 8GB RAM)

- Be used for lightweight caching, session memory, or simple agent memory

- Act as stable grounding data for LLMs building infrastructure plans

- The use cases are valid for low to high throughput workloads with 80% reads and 20% writes access pattern.

Configurations

Setting	Value
Redis Image	redis:7.2
Deployment Type	apps/v1 Deployment
Storage	Ephemeral only
Persistence	Disabled
Platform	Kubernetes
CPU/Memory Budget	2 CPUs / 2Gi RAM
Network	Exposed via NodePort

Deployment Example 1: Single-Replica Redis for Low throughput

For basic local caching and memory store use cases. (Throughput (100 ops / second))

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: redis-deployment
  labels:
    app: redis
spec:
  replicas: 1
  selector:
    matchLabels:
      app: redis
  template:
    metadata:
      labels:
        app: redis
    spec:
      containers:
        - name: redis
          image: redis:7.2
          ports:
            - containerPort: 6379
          resources:
            requests:
```

```
            memory: "512Mi"
            cpu: "250m"
          limits:
            memory: "1Gi"
            cpu: "500m"
---
apiVersion: v1
kind: Service
metadata:
  name: redis-service
spec:
  type: NodePort
  selector:
    app: redis
  ports:
    - protocol: TCP
      port: 6379
      targetPort: 6379
      nodePort: 30079
```

Deployment Example 2: High Throughput Redis (vertical Scaling for high throughput 1000 ops/second)

For read-heavy use cases or early agentic system simulation with redundancy.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: redis-deployment
  labels:
    app: redis
spec:
  replicas: 2
  selector:
    matchLabels:
      app: redis
  template:
    metadata:
      labels:
        app: redis
    spec:
      containers:
        - name: redis
          image: redis:7.2
          ports:
            - containerPort: 6379
```

```
        resources:
          requests:
            memory: "768Mi"
            cpu: "500m"
          limits:
            memory: "1.5Gi"
            cpu: "1"


---
apiVersion: v1
kind: Service
metadata:
  name: redis-service
spec:
  type: NodePort
  selector:
    app: redis
  ports:
    - protocol: TCP
      port: 6379
      targetPort: 6379
      nodePort: 30079
```