Complete Kubernetes MCP Server Testing Guide

Prerequisites Setup

1. Project Structure

```
k8s-mcp-server/
├── main.go                 # Your existing server code
├── go.mod                  # Go module file
├── go.sum                  # Go dependencies
├── docker-compose.yml      # For local K8s cluster
├── kind-config.yaml        # Kind cluster configuration
├── Makefile               # Build and test automation
├── test-scenarios/        # Test pod manifests
└── claude-config/         # Claude Desktop MCP config
    └── server-config.json
```

2. Go Module Setup

First, initialize your Go module if not already done:

```bash
# In your project directory
go mod init k8s-mcp-server

# Add required dependencies
go get github.com/mark3labs/mcp-go/mcp
go get github.com/mark3labs/mcp-go/server
go get k8s.io/client-go@latest
go get k8s.io/api@latest
go get k8s.io/apimachinery@latest
```

Docker Compose Setup

docker-compose.yml

```yaml
version: '3.8'
services:
  # Kind cluster in Docker
  kind-cluster:
    image: kindest/node:v1.27.3
    container_name: k8s-mcp-kind
    privileged: true
    ports:
      - "6443:6443"  # Kubernetes API
    volumes:
      - /var/lib/docker
    networks:
      - k8s-mcp

  # Test workloads
  nginx-healthy:
    image: nginx:latest
    container_name: test-nginx-healthy
    networks:
      - k8s-mcp
    depends_on:
      - kind-cluster

  nginx-problematic:
    image: nginx:nonexistent-tag
    container_name: test-nginx-problematic
    networks:
      - k8s-mcp
    depends_on:
      - kind-cluster

networks:
  k8s-mcp:
    driver: bridge
```

kind-config.yaml

```yaml
kind: Cluster
apiVersion: kind.x-k8s.io/v1alpha4
name: mcp-test-cluster
nodes:
- role: control-plane
  kubeadmConfigPatches:
  - |
    kind: InitConfiguration
    nodeRegistration:
      kubeletExtraArgs:
        node-labels: "ingress-ready=true"
  extraPortMappings:
  - containerPort: 80
    hostPort: 80
    protocol: TCP
  - containerPort: 443
    hostPort: 443
    protocol: TCP
- role: worker
- role: worker
```

Test Scenarios

test-scenarios/problematic-pods.yaml

yaml

```yaml
apiVersion: v1
kind: Namespace
metadata:
  name: test-problems
---
# Pod with image pull issues
apiVersion: v1
kind: Pod
metadata:
  name: bad-image-pod
  namespace: test-problems
spec:
  containers:
  - name: bad-container
    image: nonexistent/image:latest
    resources:
      requests:
        memory: "64Mi"
        cpu: "250m"
---
# Pod that crashes frequently
apiVersion: v1
kind: Pod
metadata:
  name: crash-loop-pod
  namespace: test-problems
spec:
  containers:
  - name: crash-container
    image: busybox
    command: ["sh", "-c", "sleep 10 && exit 1"]
    resources:
      requests:
        memory: "32Mi"
        cpu: "100m"
  restartPolicy: Always
---
# Pod without resources
apiVersion: v1
kind: Pod
metadata:
  name: no-resources-pod
  namespace: test-problems
spec:
  containers:
  - name: no-resources-container
```

```yaml
    image: nginx
---
# Elasticsearch-related pod for search testing
apiVersion: v1
kind: Pod
metadata:
  name: elasticsearch-test
  namespace: test-problems
  labels:
    app: elasticsearch
    environment: test
spec:
  containers:
  - name: elasticsearch
    image: elasticsearch:7.17.0
    env:
    - name: discovery.type
      value: single-node
    - name: ES_JAVA_OPTS
      value: "-Xms512m -Xmx512m"
    resources:
      requests:
        memory: "1Gi"
        cpu: "500m"
      limits:
        memory: "2Gi"
        cpu: "1000m"
```

test-scenarios/healthy-workloads.yaml

yaml

```yaml
apiVersion: v1
kind: Namespace
metadata:
  name: test-healthy
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: api-deployment
  namespace: test-healthy
spec:
  replicas: 3
  selector:
    matchLabels:
      app: api
  template:
    metadata:
      labels:
        app: api
    spec:
      containers:
      - name: api-container
        image: nginx:latest
        ports:
        - containerPort: 80
        resources:
          requests:
            memory: "128Mi"
            cpu: "100m"
          limits:
            memory: "256Mi"
            cpu: "200m"
        livenessProbe:
          httpGet:
            path: /
            port: 80
          initialDelaySeconds: 30
          periodSeconds: 10
        readinessProbe:
          httpGet:
            path: /
            port: 80
          initialDelaySeconds: 5
          periodSeconds: 5
```

Makefile

```
makefile
```

```makefile
.PHONY: setup build test clean cluster-up cluster-down deploy-test-pods

# Setup development environment
setup:
	@echo "Setting up K8s MCP Server development environment..."
	go mod tidy
	kind --version || (echo "Please install kind: https://kind.sigs.k8s.io/docs/user/qui
	kubectl version --client || (echo "Please install kubectl" && exit 1)

# Build the MCP server
build:
	@echo "Building K8s MCP Server..."
	go build -o bin/k8s-mcp-server main.go

# Create Kind cluster
cluster-up:
	@echo "Creating Kind cluster..."
	kind create cluster --config kind-config.yaml --name mcp-test-cluster
	kubectl cluster-info --context kind-mcp-test-cluster

# Deploy test scenarios
deploy-test-pods:
	@echo "Deploying test scenarios..."
	kubectl apply -f test-scenarios/problematic-pods.yaml
	kubectl apply -f test-scenarios/healthy-workloads.yaml
	@echo "Waiting for pods to be scheduled..."
	sleep 30
	kubectl get pods --all-namespaces

# Test all MCP server functions
test: build
	@echo "Testing MCP Server functions..."
	@echo "1. Testing cluster health analysis..."
	echo '{"jsonrpc": "2.0", "id": 1, "method": "tools/call", "params": {"name": "analyz

	@echo "2. Testing problematic pod detection..."
	echo '{"jsonrpc": "2.0", "id": 2, "method": "tools/call", "params": {"name": "find_p

	@echo "3. Testing pod search..."
	echo '{"jsonrpc": "2.0", "id": 3, "method": "tools/call", "params": {"name": "search

# Clean up
clean:
	kind delete cluster --name mcp-test-cluster
	rm -f bin/k8s-mcp-server
```

```
# Full test cycle
full-test: cluster-up deploy-test-pods test

# Claude Desktop integration test
claude-test: build
	@echo "Testing Claude Desktop integration..."
	@echo "Make sure to add the server config to Claude Desktop first!"
	@echo "Server binary ready at: $(PWD)/bin/k8s-mcp-server"
```

Claude Desktop Configuration

Claude Desktop MCP Server Config

Add this to your Claude Desktop configuration file:

macOS: `~/Library/Application Support/Claude/claude_desktop_config.json` **Windows**: `%APPDATA%/Claude/claude_desktop_config.json`

```json
{
  "mcpServers": {
    "k8s-diagnostics": {
      "command": "/path/to/your/project/bin/k8s-mcp-server",
      "env": {
        "KUBECONFIG": "/path/to/your/.kube/config"
      }
    }
  }
}
```

Step-by-Step Testing Process

1. Environment Setup

```bash
# Clone/create your project directory
mkdir k8s-mcp-server && cd k8s-mcp-server

# Copy your main.go file
# Create the additional files from this guide

# Initialize and setup
make setup
```

2. Build and Test Locally

```bash
# Build the server
make build

# Create test cluster
make cluster-up

# Deploy test scenarios
make deploy-test-pods

# Verify cluster state
kubectl get pods --all-namespaces
kubectl get nodes
```

3. Test MCP Server Functions

```bash
# Test each function individually
./bin/k8s-mcp-server &
SERVER_PID=$!

# Test via JSON-RPC (in another terminal)
echo '{"jsonrpc": "2.0", "id": 1, "method": "tools/call", "params": {"name": "quick_tr

kill $SERVER_PID
```

4. Claude Desktop Integration

```bash
# Build for Claude Desktop
make build

# Update Claude Desktop config with correct paths
# Restart Claude Desktop

# Test queries in Claude Desktop:
# "What's the health of my Kubernetes cluster?"
# "Find all problematic pods"
# "Show me pods with high restart counts"
# "Search for elasticsearch pods"
```

Test Queries for Claude Desktop

Once configured, try these natural language queries:

Basic Health Checks

- "What's wrong with my Kubernetes cluster?"
- "Give me a quick health overview"
- "Analyze my cluster health"

Problem Detection

- "Find all failing pods in my cluster"
- "Show me pods that are restarting frequently"
- "What pods have image pull issues?"
- "List all pods with problems"

Targeted Searches

- "Find all elasticsearch-related pods"
- "Show me anything related to 'api' in the test-healthy namespace"
- "Search for pods with 'crash' in the name"

Resource Analysis

- "Which pods are consuming the most resources?"
- "Show me pods without resource limits"
- "Get resource usage overview"

Detailed Diagnostics

- "Diagnose the bad-image-pod in test-problems namespace"
- "Analyze logs for elasticsearch-test pod"
- "Get workload recommendations for test-healthy namespace"

Troubleshooting

Common Issues

1. **Kind cluster not accessible**

   bash

   ```bash
   kubectl cluster-info --context kind-mcp-test-cluster
   ```

2. **MCP server not connecting**
   - Check file paths in Claude Desktop config

   - Verify binary permissions: `chmod +x bin/k8s-mcp-server`
   - Check KUBECONFIG environment variable

3. **Test pods not appearing**

   ```bash
   kubectl get pods --all-namespaces
   kubectl describe pod bad-image-pod -n test-problems
   ```

4. **Claude Desktop not recognizing server**
   - Restart Claude Desktop after config changes
   - Check server logs
   - Verify JSON config syntax

Expected Test Results

After setup, you should see:

- **Healthy pods**: api-deployment pods running normally
- **Problematic pods**: bad-image-pod (ImagePullBackOff), crash-loop-pod (CrashLoopBackOff)
- **Resource issues**: no-resources-pod flagged for missing limits
- **Search capabilities**: Finding elasticsearch-test by pattern matching

This gives you a comprehensive test environment to validate all MCP server capabilities!