Kubernetes Cluster Health Checker: Beginner-Friendly End-to-End Guide

This step-by-step guide will help you set up, configure, test, and submit a simple Kubernetes cluster health–checker project. No prior experience is assumed—each step includes detailed explanations and commands. You'll also learn how to version-control the project and push it to your GitHub account.

---

1. Introduction and Objectives

What You Will Build:

- A sample microservice deployment on Minikube.
- Health checks (liveness/readiness probes) and self-healing configuration.
- Basic monitoring with Prometheus and Grafana.
- A simple alert rule to detect an unhealthy pod.
- A dashboard to visualize pod metrics.

Why These Steps Matter:

- **Self-healing probes** ensure automatic restart on failure.
- **Monitoring & alerting** let you detect and respond to issues quickly.
- **Version control** (Git + GitHub) tracks changes and shares your work.

---

2. Prerequisites and Tool Installation

Tools You'll Need:

1. **Minikube** (local Kubernetes cluster)
2. **kubectl** (Kubernetes CLI)
3. **Helm** (Kubernetes package manager)
4. **Git** (version control)
5. **GitHub account** (to host your repository)

2.1 Install Minikube

- Follow the official guide: https://minikube.sigs.k8s.io/docs/start/
- After installation, verify:

```
minikube version
```

2.2 Install kubectl

- Follow: https://kubernetes.io/docs/tasks/tools/
- Verify:

```
kubectl version --client
```

2.3 Install Helm

- Follow: https://helm.sh/docs/intro/install/
- Verify:

```
helm version
```

2.4 Install Git

- Windows: https://git-scm.com/download/win
- Mac/Linux: use your package manager (e.g., `sudo apt install git`)
- Verify:

```
git --version
```

---

3. Project Setup and Directory Structure

In this step, you'll create a clear folder layout so all your files stay organized. Here’s the recommended structure:

```
k8s-health-checker/                  # Project root
├── app/                             # (Optional) Your application code or sample
scripts
│   └── README.md                    # Overview of the sample app
├── kubernetes/                      # Kubernetes configurations
│   ├── base/                        # Core manifests
│   │   ├── deployment.yaml          # Deployment for echo-server
│   │   └── service.yaml             # Service exposing echo-server
│   └── monitoring/                  # Monitoring and alerting configs
│       ├── alert.yaml               # PrometheusRule definitions
│       └── README.md                # Notes on monitoring setup
├── charts/                          # (Optional) Helm chart if you package it
later
├── .gitignore                       # Git ignore rules
├── README.md                        # Project overview and instructions
└── LICENSE                          # (Optional) License for your project
```

Steps to set up:

1. Open a terminal and navigate to where you want the project:

```
cd ~/projects
mkdir k8s-health-checker && cd k8s-health-checker
```

2. If you have the ZIP, unzip it here:

```
unzip /path/to/CapstoneProject-k8s-health-checker.zip -d .
```

3. Otherwise, create folders manually:

```
mkdir app kubernetes kubernetes/base kubernetes/monitoring charts
touch .gitignore LICENSE README.md app/README.md kubernetes/monitoring/
README.md
```

4. Open this folder in your code editor (e.g., VS Code):

```
code .
```

---

4. Initialize Git and Create GitHub Repository Initialize Git and Create GitHub Repository

1. **Initialize Git** locally:

```
cd ~/projects/k8s-health-checker
git init
git add .
git commit -m "Initial project structure"
```

2. **Create a new GitHub repository** named `k8s-health-checker`:
3. Visit https://github.com/new
4. Set **Repository name** to `k8s-health-checker`
5. Leave **Initialize with README** unchecked (you already have one).
6. **Add remote and push**:

```
git remote add origin https://github.com/<your-username>/k8s-health-
checker.git
```

```
git branch -M main
git push -u origin main
```

---

5. Start Minikube

1. Launch a local cluster:

```
minikube start --driver=docker
```

2. Verify node status:

```
kubectl get nodes
```

---

6. Deploy Sample Application with Health Probes

1. **Create a simple HTTP server** (optional):
2. You can use any Docker image (e.g., `hashicorp/http-echo`).
3. **Write a Deployment manifest** (`kubernetes/base/deployment.yaml`):

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: echo-server
spec:
  replicas: 2
  selector:
    matchLabels:
      app: echo
  template:
    metadata:
      labels:
        app: echo
    spec:
      containers:
      - name: http-echo
        image: hashicorp/http-echo:0.2.3
        args:
        - "-text=Hello Kubernetes"
        ports:
        - containerPort: 5678
        livenessProbe:
          httpGet:
            path: /
            port: 5678
          initialDelaySeconds: 5
          periodSeconds: 10
        readinessProbe:
          httpGet:
            path: /
            port: 5678
          initialDelaySeconds: 5
          periodSeconds: 5
```

4. **Apply manifests**:

```
kubectl apply -f kubernetes/base/deployment.yaml
```

5. **Check pods**:

```
kubectl get pods -l app=echo
```

---

7. Add Service for Access

1. **Service manifest** (`kubernetes/base/service.yaml`):

```
apiVersion: v1
kind: Service
metadata:
  name: echo-service
spec:
  selector:
    app: echo
  ports:
  - port: 80
    targetPort: 5678
```

2. **Apply and test**:

```
kubectl apply -f kubernetes/base/service.yaml
minikube service echo-service --url
```

3. Open the printed URL in your browser; you should see **Hello Kubernetes**.

---

8. Setup Monitoring with Helm

1. **Add Prometheus & Grafana chart repo**:

```
helm repo add prometheus-community https://prometheus-community.github.io/
helm-charts
helm repo update
```

2. **Install Prometheus**:

```
helm install prometheus prometheus-community/prometheus --namespace
monitoring --create-namespace
```

3. **Install Grafana**:

```
helm install grafana prometheus-community/grafana --namespace monitoring
```

4. **Access Grafana**:

```
kubectl port-forward svc/grafana 3000:80 -n monitoring
```

5. In your browser, go to `http://localhost:3000`
6. Default login: `admin` / `prom-operator`

---

9. Configure Alerting

1. **Create an AlertRule** (`kubernetes/monitoring/alert.yaml`):

```
apiVersion: monitoring.coreos.com/v1
kind: PrometheusRule
metadata:
  name: echo-alerts
spec:
  groups:
  - name: echo.rules
    rules:
    - alert: PodCrashLooping
      expr:
rate(kube_pod_container_status_restarts_total{namespace="default"}[5m]) > 0
      for: 2m
      labels:
        severity: warning
      annotations:
        summary: "Pod is restarting frequently"
```

2. **Apply rule**:

```
kubectl apply -f kubernetes/monitoring/alert.yaml
```

3. **Trigger a failure** by editing the Deployment to an invalid image and watch the alert fire in Grafana's Alerting section.

---

10. Push Changes and Final Submission

1. **Commit all your YAML files**:

```
git add kubernetes/ base/ monitoring/ README.md
git commit -m "Completed health checker with monitoring and alerts"
git push
```

2. **Verify on GitHub**: Go to `https://github.com/<your-username>/k8s-health-checker` to see your files.

Congratulations! You've built a complete Kubernetes health-checking and monitoring solution from scratch, tested it end-to-end, and submitted it to GitHub.