**Part 1: Monitoring Python Application**

**Method 1: Add Prometheus Client Library to Python App**

**Install the Prometheus client library**:[[1]](#fn1)[[2]](#fn2)

# requirements.txt  
prometheus-client==0.20.0  
# or for Flask apps  
prometheus-flask-exporter==0.23.0

**For a Flask application**:[[3]](#fn3)[[2]](#fn2)

from flask import Flask  
from prometheus\_flask\_exporter import PrometheusMetrics  
  
app = Flask(\_\_name\_\_)  
metrics = PrometheusMetrics(app)  
  
@app.route('/')  
def hello():  
 return "Hello World!"  
  
@app.route('/health')  
def health():  
 return "OK"  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app.run(host='0.0.0.0', port=8080)

This automatically exposes metrics at http://your-app:8080/metrics.[[2]](#fn2)[[1]](#fn1)

**For a generic Python application**:[[1]](#fn1)

from prometheus\_client import Counter, Gauge, Histogram, start\_http\_server  
import time  
  
# Define metrics  
request\_count = Counter('app\_requests\_total', 'Total app requests')  
active\_users = Gauge('app\_active\_users', 'Currently active users')  
request\_duration = Histogram('app\_request\_duration\_seconds', 'Request duration')  
  
# Start Prometheus metrics server on port 8000  
start\_http\_server(8000)  
  
# Your application logic  
while True:  
 request\_count.inc()  
 active\_users.set(42)  
 time.sleep(10)

**Deploy Python App with Metrics**

**Deployment manifest** (python-app.yaml):[[3]](#fn3)

apiVersion: apps/v1  
kind: Deployment  
metadata:  
 name: python-app  
 namespace: default  
 labels:  
 app: python-app  
spec:  
 replicas: 1  
 selector:  
 matchLabels:  
 app: python-app  
 template:  
 metadata:  
 labels:  
 app: python-app  
 spec:  
 containers:  
 - name: python-app  
 image: your-registry/python-app:latest  
 ports:  
 - containerPort: 8080  
 name: http  
---  
apiVersion: v1  
kind: Service  
metadata:  
 name: python-app-service  
 namespace: default  
 labels:  
 app: python-app  
spec:  
 selector:  
 app: python-app  
 ports:  
 - port: 8080  
 targetPort: 8080  
 name: http

**Create ServiceMonitor for Python App**

**ServiceMonitor manifest** (python-servicemonitor.yaml):[[4]](#fn4)[[3]](#fn3)

apiVersion: monitoring.coreos.com/v1  
kind: ServiceMonitor  
metadata:  
 name: python-app-monitor  
 namespace: monitoring  
 labels:  
 release: kube-prometheus-stack  
spec:  
 selector:  
 matchLabels:  
 app: python-app  
 namespaceSelector:  
 matchNames:  
 - default  
 endpoints:  
 - port: http  
 path: /metrics  
 interval: 30s

Apply it:

kubectl apply -f python-servicemonitor.yaml

**Part 2: Monitoring Redis**

**Deploy Redis Exporter**

Redis doesn't expose Prometheus metrics natively, so you need the **Redis Exporter**.[[5]](#fn5)[[6]](#fn6)[[7]](#fn7)

**Step 1: Create Redis Auth Secret** (if Redis has authentication):[[5]](#fn5)

kubectl create secret generic redis-exporter-auth \  
 --from-literal=password='your-redis-password' \  
 -n default

**Step 2: Deploy Redis Exporter** (redis-exporter.yaml):[[6]](#fn6)[[5]](#fn5)

apiVersion: apps/v1  
kind: Deployment  
metadata:  
 name: redis-exporter  
 namespace: default  
 labels:  
 app: redis-exporter  
spec:  
 replicas: 1  
 selector:  
 matchLabels:  
 app: redis-exporter  
 template:  
 metadata:  
 labels:  
 app: redis-exporter  
 annotations:  
 prometheus.io/scrape: "true"  
 prometheus.io/port: "9121"  
 spec:  
 containers:  
 - name: redis-exporter  
 image: oliver006/redis\_exporter:latest  
 ports:  
 - containerPort: 9121  
 name: metrics  
 env:  
 - name: REDIS\_ADDR  
 value: "redis-service:6379" # Your Redis service name  
 - name: REDIS\_PASSWORD  
 valueFrom:  
 secretKeyRef:  
 name: redis-exporter-auth  
 key: password  
 optional: true  
 resources:  
 limits:  
 memory: "128Mi"  
 cpu: "100m"  
---  
apiVersion: v1  
kind: Service  
metadata:  
 name: redis-exporter-service  
 namespace: default  
 labels:  
 app: redis-exporter  
spec:  
 selector:  
 app: redis-exporter  
 ports:  
 - port: 9121  
 targetPort: 9121  
 name: metrics

Apply it:

kubectl apply -f redis-exporter.yaml

**Create ServiceMonitor for Redis**

**ServiceMonitor manifest** (redis-servicemonitor.yaml):[[8]](#fn8)[[6]](#fn6)

apiVersion: monitoring.coreos.com/v1  
kind: ServiceMonitor  
metadata:  
 name: redis-exporter-monitor  
 namespace: monitoring  
 labels:  
 release: kube-prometheus-stack  
spec:  
 selector:  
 matchLabels:  
 app: redis-exporter  
 namespaceSelector:  
 matchNames:  
 - default  
 endpoints:  
 - port: metrics  
 interval: 30s  
 path: /metrics

Apply it:

kubectl apply -f redis-servicemonitor.yaml

**Part 3: Verify Everything is Working**

**Check ServiceMonitors**

kubectl get servicemonitor -n monitoring

You should see:

* python-app-monitor
* redis-exporter-monitor

**Verify Prometheus Targets**

# Access Prometheus UI  
kubectl port-forward -n monitoring svc/kube-prometheus-stack-prometheus 9090:9090

Open browser: http://localhost:9090/targets

Look for your targets:

* serviceMonitor/monitoring/python-app-monitor/0
* serviceMonitor/monitoring/redis-exporter-monitor/0

Both should show status **UP**.[[6]](#fn6)[[3]](#fn3)

**Test Metrics in Prometheus**

In Prometheus UI, try these queries:[[7]](#fn7)[[9]](#fn9)

**Python app metrics:**

# Request count  
rate(flask\_http\_request\_total[5m])  
  
# Request duration  
histogram\_quantile(0.95, rate(flask\_http\_request\_duration\_seconds\_bucket[5m]))

**Redis metrics:**

# Redis uptime  
redis\_uptime\_in\_seconds  
  
# Connected clients  
redis\_connected\_clients  
  
# Memory usage  
redis\_memory\_used\_bytes  
  
# Commands per second  
rate(redis\_commands\_processed\_total[5m])  
  
# Keyspace hits ratio  
redis\_keyspace\_hits\_total / (redis\_keyspace\_hits\_total + redis\_keyspace\_misses\_total)

**Part 4: Create Grafana Dashboards**

**Import Pre-built Redis Dashboard**

1. Go to Grafana (access via NodePort)
2. Click **+ → Import**
3. Enter Dashboard ID: **11835** (Redis Dashboard for Prometheus)[[6]](#fn6)
4. Select Prometheus data source
5. Click **Import**

**Create Custom Python App Dashboard**

1. **+ → Dashboard → Add visualization**
2. Select Prometheus data source
3. Add panels with queries:

**Panel 1: Request Rate**

rate(flask\_http\_request\_total{namespace="default", pod=~"python-app.\*"}[5m])

**Panel 2: Response Time (95th percentile)**

histogram\_quantile(0.95, rate(flask\_http\_request\_duration\_seconds\_bucket{namespace="default"}[5m]))

**Panel 3: Error Rate**

rate(flask\_http\_request\_total{namespace="default", status=~"5.."}[5m])

**Troubleshooting**

**ServiceMonitor Not Working**

Check if ServiceMonitor label matches Prometheus selector:[[10]](#fn10)[[3]](#fn3)

# Check Prometheus ServiceMonitor selector  
kubectl get prometheus -n monitoring kube-prometheus-stack-prometheus -o yaml | grep serviceMonitorSelector -A 5

Your ServiceMonitor needs label: release: kube-prometheus-stack.[[3]](#fn3)

**No Metrics Showing**

Test metrics endpoint directly:

# For Python app  
kubectl port-forward -n default svc/python-app-service 8080:8080  
curl http://localhost:8080/metrics  
  
# For Redis exporter  
kubectl port-forward -n default svc/redis-exporter-service 9121:9121  
curl http://localhost:9121/metrics

You should see Prometheus-formatted metrics.[[1]](#fn1)[[5]](#fn5)

This setup gives you complete monitoring of both Python applications and Redis databases in your Kubernetes cluster![[3]](#fn3)[[6]](#fn6)

⁂

1. <https://betterstack.com/community/guides/monitoring/prometheus-python-metrics/>

1. <https://blog.viktoradam.net/2020/05/11/prometheus-flask-exporter/>

1. <https://devopsvoyager.hashnode.dev/servicemonitor-in-prometheus>

1. <https://signoz.io/guides/how-to-monitor-custom-kubernetes-pod-metrics-using-prometheus/>

1. <https://www.sysdig.com/blog/redis-prometheus>

1. <https://dev.to/rslim087a/monitoring-redis-with-prometheus-and-grafana-56pk>

1. <https://logz.io/blog/how-to-monitor-redis-with-prometheus/>

1. <https://redis.io/docs/latest/operate/kubernetes/re-clusters/connect-prometheus-operator/>

1. <https://signoz.io/guides/how-to-get-number-of-pods-running-in-prometheus/>

1. <https://github.com/prometheus-operator/kube-prometheus/issues/1069>

1. <https://devopscube.com/setup-prometheus-monitoring-on-kubernetes/>

1. <https://stackoverflow.com/questions/59656925/setting-up-prometheus-with-python-in-kubernetes-cluster>

1. <https://prometheus.io>

1. <https://www.plural.sh/blog/prometheus-kubernetes-monitoring-guide/>

1. <https://python.plainenglish.io/how-to-monitor-python-application-with-kafka-prometheus-9c4199bb9e5a>

1. <https://stackoverflow.com/questions/77145578/configuration-for-kubernetes-flask-and-prometheus>

1. <https://github.com/prometheus-operator/kube-prometheus>

1. <https://github.com/oliver006/redis_exporter>

1. <https://dev.to/camptocamp-ops/integrate-an-application-with-prometheus-operator-and-package-with-a-helm-chart-1159>

1. <https://stackoverflow.com/questions/76684401/k8s-how-to-enable-metric-collecting-for-redis-prometheus>

1. <https://github.com/rycus86/prometheus_flask_exporter>