ZopAI Config Optimizer Report

Generated on: 2025-08-18 21:41:23

Config Health Score: 50/100

Detected Warnings:

WARNING! runAsRoot detected -> set runAsNonRoot=true, runAsUser=1000

WARNING! Resources: Missing CPU/memory -> added default requests/limits

WARNING! Missing livenessProbe -> added HTTP probe

WARNING! Missing readinessProbe -> added HTTP probe

WARNING! RBAC disabled -> set rbac.create=true

AI Suggestions:

This Kubernetes configuration has several significant security and stability risks. Let's address them with best practices:

1. `securityContext: runAsUser: 0` (Running as root):

* **Best Practice:** **Never run containers as root.** This is a major security vulnerability. If a container is compromised, the attacker gains root access to the entire host node.

* **Solution:** Assign a non-root user ID. You need to create a user inside the container image with appropriate permissions. This requires modifying your Dockerfile.

```yaml
securityContext:
runAsUser: 1000 # Replace with a non-root UID from your image
runAsGroup: 1000 # Ideally match the group ID as well
```

* **Important:** Ensure the user you choose exists in your container image *before* building it. The UID/GID must exist within the image's filesystem. If you use a base image with a pre-existing non-root user, you can use that UID.

2. Missing Resource Limits:

* **Best Practice:** Always define `requests` and `limits` for CPU and memory. This prevents runaway containers from consuming all resources and impacting other pods. Without limits, a misbehaving container can crash the entire node.

* **Solution:** Add resource limits to your deployment spec:

```yaml
```

resources:
requests:
cpu: 100m
memory: 256Mi
limits:
cpu: 500m
memory: 1Gi
```
Adjust the values according to your application's needs.

**3. Missing Probes (Liveness and Readiness):**

* **Best Practice:** Implement liveness and readiness probes. Liveness probes check if the container is still running and healthy. Readiness probes check if the container is ready to accept traffic. Without probes, Kubernetes cannot effectively manage the health of your application.

* **Solution:** Add liveness and readiness probes using `exec`, `httpGet`, or `tcpSocket`. Example using `httpGet`:

```yaml
livenessProbe:
httpGet:
path: /healthz
port: 8080
readinessProbe:
httpGet:
path: /ready
port: 8080
```
You'll need to create endpoints `/healthz` and `/ready` in your application. The port should match the port your application exposes.

4. `rbac: create: false`:

* **Best Practice:** Use RBAC to grant only necessary permissions. While `create: false` might seem restrictive, it depends heavily on your application's needs. If your application requires creating resources (e.g., ConfigMaps, Secrets), you'll need to define appropriate RBAC roles and role bindings. Otherwise this is acceptable if the pod doesn't need to create any Kubernetes resources.

* **Solution:** If you need to create resources, define a Role and RoleBinding granting the necessary permissions. If not, this setting is acceptable, but reconsider the scope of the application.

Complete Example (Illustrative):

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
name: my-app
spec:
selector:
matchLabels:
app: my-app
template:
metadata:
labels:
```
```

app: my-app
spec:
containers:
- name: my-app-container
image: my-image:latest
securityContext:
runAsUser: 1000
runAsGroup: 1000
resources:
requests:
cpu: 100m
memory: 256Mi
limits:
cpu: 500m
memory: 1Gi
livenessProbe:
httpGet:
path: /healthz
port: 8080
readinessProbe:
httpGet:
path: /ready
port: 8080
```

Remember to replace placeholders like `my-image:latest`, `1000`, `8080`, `/healthz`, and `/ready` with your actual values. Always thoroughly test your changes after implementing these best practices.

## Optimized YAML:

```
livenessProbe:
  httpGet:
    path: /
    port: 80
  initialDelaySeconds: 10
  periodSeconds: 5
rbac:
  create: true
readinessProbe:
  httpGet:
    path: /
    port: 80
  initialDelaySeconds: 10
  periodSeconds: 5
resources:
  limits:
    cpu: 500m
    memory: 512Mi
  requests:
    cpu: 250m
    memory: 256Mi
securityContext:
  runAsNonRoot: true
```
```

```
runAsUser: 1000
```