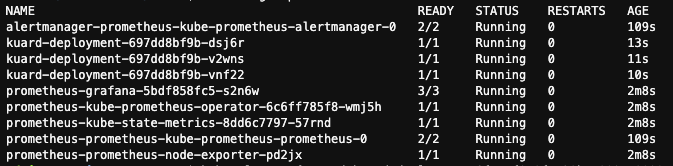
Documentation

# Install of Prometheus and Grafana:

Refer to the challenge.sh.

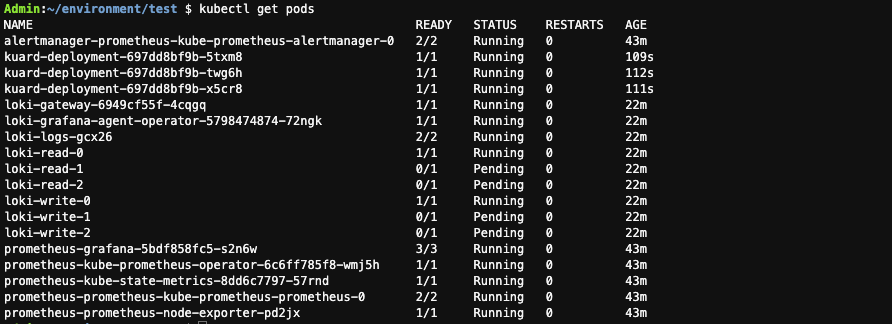


# Wie würde eine Logging-Lösung aussehen?

In dieser Aufgabe wurde Loki als Logging-Lösung verwendet. Loki sammelt Logs und Metriken in ein Loki-StatefulSet. Loki kann dabei als Datenbank gesehen werden, die Metriken an einem zentralen Ort in einem Kubernetes-Cluster aggregiert.

*Funktionsweise:*

Durch die Installation von Prometheus und Loki läuft der Promtail-Agent auf den Worker-Nodes. Dieser erkennt Metriken und pusht diese in das Loki-StatefulSet. In der folgenden Abbildung sind die generierten Pods von Loki & Grafana & Prometheus zu erkennen:



In meiner Deployment-File habe ich drei Pods der Kuard-Application bereitgestellt. Da ich lokal jedoch nur ein Volume-Claim zur Verfügung habe, können nur Logs von einem Kuard-Pod aggregiert werden.

# Wie kann Autoscaling in einem Cluster erreicht werden?

Siehe befehl in challenge.sh.

Die Pods müssen hinter einem Service bereitgestellt werden. Dann kann ein Autoscaler von Kubernetes bereitgestellt werden, der ggf. die Pods aufskaliert. In dem Beispiel von challenge.sh wird eine Target-Tracking-Policy verwendet. Sobald die Pods eine CPU-Usage von 80% haben, werden die Pods aufskaliert. In der Abbildung ist die Autoscaling-Komponente zu erkennen:



# Wie kann man Fehler im Templating erkennen?

Als Tool zur Überprüfung der Yaml-Datei wird kubeval verwendet. Für die Installation kann man in die challenge.sh reinschauen.



# Welche Möglichkeiten gibt es ein Deployment eine frühere Version zurückzusetzen?

In der challenge.sh erkennt man die Befehle für das Rücksetzen eines Deployments durch ein sogennantes Rollback. Dabei kann ein Deployment auf eine frühere Version zurückgesetzt werden.

# Welche Metriken?

Man kann den memory überprüfen (heap alloc,heap idle), readiness probes durchführen etc.

# Welches CI/CD Tool?

ArgoCD setzt sich gerade im Kubernetes umfeld durch. Vorteile: Blue green deployment, automatisierte Rollbacks, weighted traffic shifting etc.

# Ingress Controller?

Ich habe meistens Elastic Balancer in Kombination mit Istio verwendet, um meinen Service öffentlich erreichbar zu machen.

# Secrets speichern?

Ich persönlich habe immer den AWS Secrets Manager mit EKS verwendet.

# Secrets erkennen.

Ich kann nur eine Lösung in AWS vorschlagen.

CodeGuru ist neuer Service von AWS der Secrets im Quellcode erkennt. Er erkennt nicht nur das, sonder auch fehlerhafter Code -> Falsche Verwendung von Krypto-Bibs etc.

Falls Secret erkannt wird: https://aws.amazon.com/blogs/aws/codeguru-reviewer-secrets-detector-identify-hardcoded-secrets/