What did we do?

In this final lesson, we will see everything which we covered in this course.

WE'LL COVER THE FOLLOWING

Scaled pods and cluster nodes

We explored quite a few topics that are beyond “normal” Kubernetes usage. We learned how to scale Pods using `HorizontalPodAutoscaler`. We saw that scaling Pods does not provide enough benefits if we cannot scale cluster nodes as well. We explored how to do just that using `Cluster Autoscaler`. Unfortunately, it is currently available only for AWS, GKE, and AKS.

Gathered Metrics

While scaling Pods and nodes is essential, we had to gather metrics as well. They give us insights into the behavior of our clusters and applications running inside it. For that, we adopted `Prometheus`. More importantly, we saw how we can leverage `Alertmanager` to create notifications that will alert us when there is an issue, instead of staring at a screen waiting for a graph to reach a “red line”.

Instrumented our applications

We learned that gathering metrics from exporters might not be enough, so we instrumented our applications as a way of providing lower-level metrics that give us deep insights into the state of our applications.

Explored Custom Metrics

We also explored `HorizontalPodAutoscaler`'s ability to use custom metrics. We hooked it into `Prometheus` thus extending scaling thresholds to almost any formula we could imagine.

Connected `Prometheus` to Grafana

Given that we collect metrics in `Prometheus` and that it doesn't provide dashboarding functionality, we connected it to Grafana. During the process, we explored ways that dashboards can be made more useful than traditional "pretty colors" many are instinctively drawn to.

Explored Centralized Logging

Finally, we discussed the need for centralized logging as well as a few tools that might help us debug issues discovered through alerts. For that, we evaluated **Papertrail**, **AWS CloudWatch**, **GCP Stackdriver**, **Azure Log Analytics**, and the **Elasticsearch-Fluentd-Kibana (EFK) stack**.

Robust and reliable clusters

We went beyond common Kubernetes operations and managed to make our clusters more robust and more reliable. We made a step forward towards mostly autonomous self-adaptive systems that require human intervention only in exceptional cases that cannot be resolved automatically. When we do need to intervene, we'll do that equipped with all the necessary information required to quickly deduce the cause of the issue and perform corrective actions that will restore our cluster to the desired state.