OTH Regensburg

Faculty: Maths / Computer Science

FWPM-Module: KICC – Introduction to Cloud Computing

Summer Semester 2021

Lecturer: Mr. Maximilian Schön

Logging & Monitoring

of a Kubernetes Cluster and its running applications – implemented on AWS

Submission date: 28.04.2021

Author: Daniel Gruber

E-Mail: daniel.gruber@st.oth-regensburg.de

Course: Technical Computer Science

Matriculation number: 3214109

Content

1. Description of task

The whole Lab exercise is implemented on Amazon Web Services (abbreviation: AWS).

The main task of this lab exercise was to set up a Kubernetes Cluster which is able to aggregate and visualize logs. Therefore, additional applications like Prometheus and Grafana are needed for monitoring and e.g. CloudWatch as a service from AWS for logging. As follows a specification of this exercise was to configure several dashboards representing the state of the cluster and its running applications. (Kubernetes Dashboard, Prometheus, Grafana)
Additionally a simple application should be used, which produces log entries and could be everything from a simple "Hello World" to multiple microservices. In my case several demo applications are deployed and one specific app called "mywebping".

2. Used Services

2.1 AWS Cost Management

As one of the big advantages of going for the Cloud is the "only pay for use" model. The AWS Cost Management offers several possibilities and features to have an overview of outgoings and budgets. As some services I used were charged few budget reports offered a good overall picture of my expenditures.

2.2 Identity and Access Management

With AWS Identity Access Management (IAM) management of access to AWS Services and resources is made securely. Using IAM offers creation and management of users and groups, to which permissions and policies can be applied to allow or deny access to AWS resources.

2.3 Cloud9 (+ EC2 and EBS)

AWS Cloud9 is a cloud-based integrated development environment (IDE) that makes writing, running and managing other AWS services very easy. Cloud9 includes editor, debugger, terminal and several features like previews of applications forwarded to dedicated ports. On creation Cloud9 offers already tools for several programming languages and most importantly for accomplishing this exercise many package managers and command tools are preinstalled like eksctl. As Cloud9 is based on an EC2 (Elastic Compute Cloud) instance, it has an EBS (Elastic Block Storage) volume. Cloud9 stops automatically if no action is recognized for more than 30 minutes.

2.4 Elastic Kubernetes Service (EKS)

Amazon Elastic Kubernetes Service (EKS) makes starting, running and scaling Kubernetes applications in the AWS Cloud or non-premises uncomplicated, as several Kubernetes functionalities are managed by Amazon EKS. The whole control plane is managed by EKS and as "data plane" deployment of managed node worker groups were necessary or alternatively using the AWS Service Fargate is possible.

2.5 Elastic Container Registry (ECR)

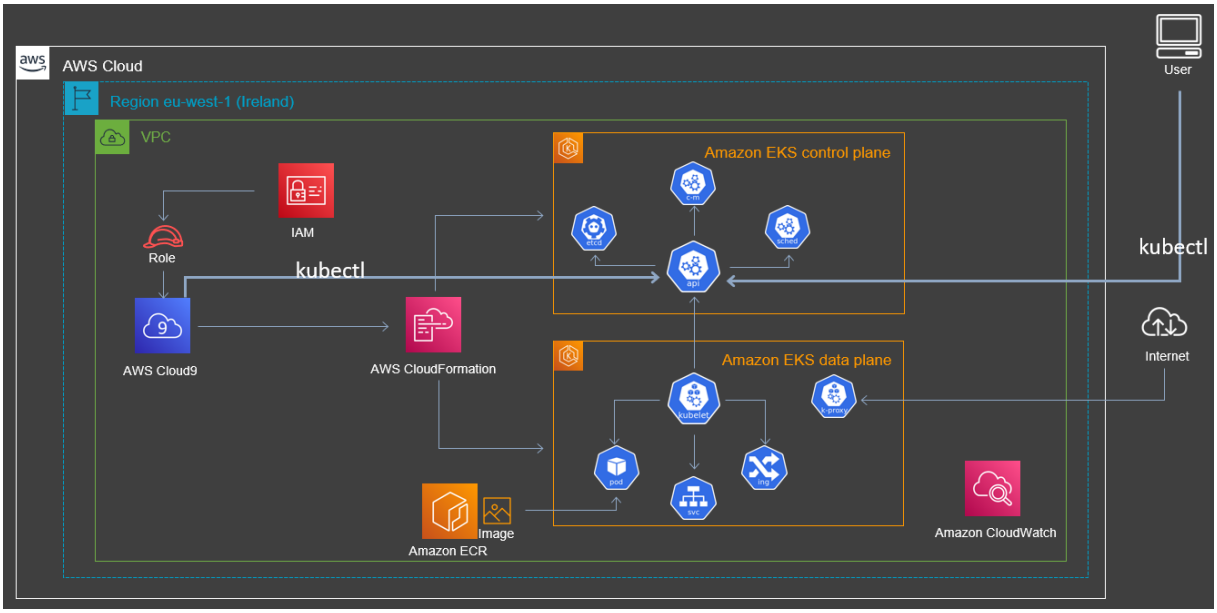
Amazon Elastic Container Registry (ECR) is a fully managed container registry that makes it easy to store, manage, share and deploy container images anywhere. Container images are hosted in a highly available and high-performance architecture, which guarantees reliability. Container Software can be shared privately within your organization or publicly worldwide. ECR works with Amazon EKS and simplifies getting started deploying containers and utilizing Fargate it provides one-click deployments.

2.6 CloudWatch

Amazon CloudWatch is a monitoring and observability service providing data and actionable insights for monitoring applications, responding to performance changes and getting a unified view of operational health. CloudWatch collects monitoring and operational data in the form of logs, metrics, and events, offering a unified view of AWS resources, applications, and services running on AWS. It can be used for detecting anomalous behaviour, setting alarms, visualizing logs and metrics and discovering insights of for example running applications. (in my case container insights)

3. My Kubernetes Cluster “myEKS”

3.1 Architectural Overview



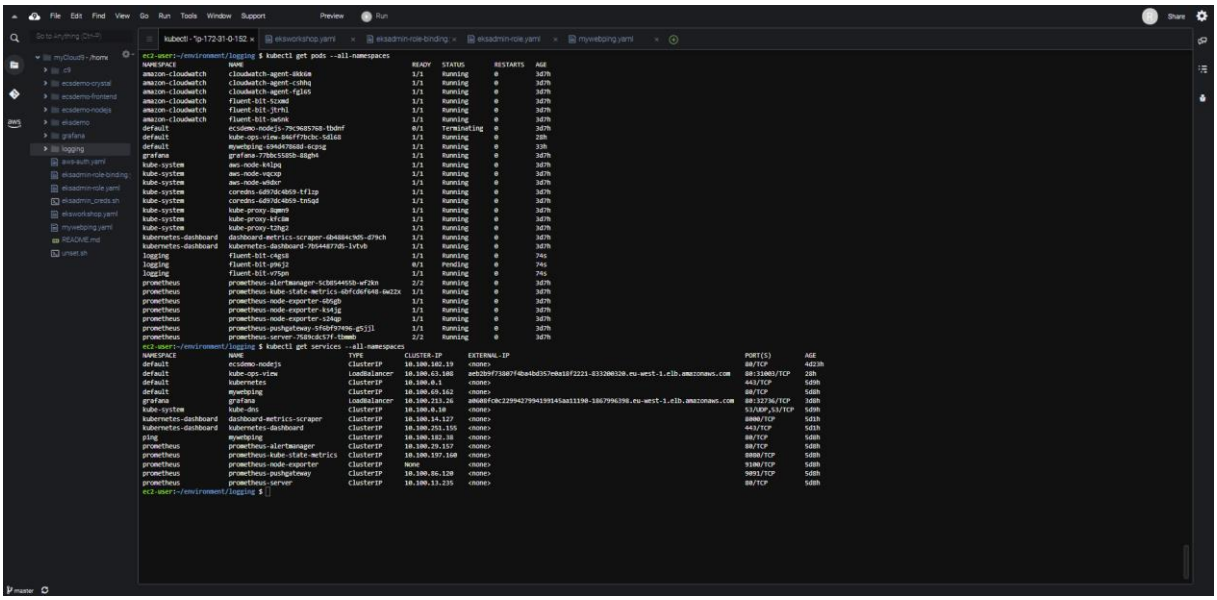
Screenshot 1: Architectural Overview

The image above shows the rough concept how the kubernetes cluster is created, managed and implemented. Further on all core AWS Services and their coherences to other services are represented. In the following sections the creation workflow and all services, features and applications used are described in its purposes and how they are implemented. Moreover it will explained why they are used and which advantages these services bring with them but this documentation doesn’t contain a step -for-step guide how this was created. This can be viewed in the videos provided in the folder 03_videos and links provided in this documentation.

3.2 Setting up my Workspace (using 2.2 IAM and 2.3 Cloud9)

First step concerns about creating an environment in Cloud9. Therefore a Cloud9 Instance was created. For managing an EKS cluster tools like kubectl are installed and awscli was updated. Additionally the AWS Load Balancer Controller version was set in the bash profile.
As next steps an IAM role with administrator rights was created and attached to the EC2 instance of the Cloud9 environment.
In the image below an example for the Look and Feel of an Cloud9 Environment is given. Also two example for working with the kubectl command are shown.

The next chapter describes how my kubernetes cluster was created which basically relies on working on Cloud9 using the command eksctl.



Screenshot 2: Cloud9 environment

3.3 Creating myEKS kubernetes cluster (using 2.4 EKS)

The kubernetes cluster on AWS is created with the command tool eksctl by entering following command into the terminal:

$eksctl create cluster -f myEKS_clustercreation.yaml

>>> see folder 02_yaml_json\myEKS_clustercreation.yaml

In this yaml file metadate like name, region, version and availability Zones of the Kubernetes cluster are specified.
Also a managed Node-Group of two t3.small instances are specified. Later on I updated a few setting options for the autoscaling group of my managed node group. Also I experienced with Instance refreshments. See >>> 03_videos\01_Cloud9.mkv from minute ~3.30 onwards.

3.4 Deployment & Pods (using 2.5 ECR)

For testing and getting familiar with running pods on my kubernetes cluster I deployed several demo apps by applying yaml files with kubectl with this command:

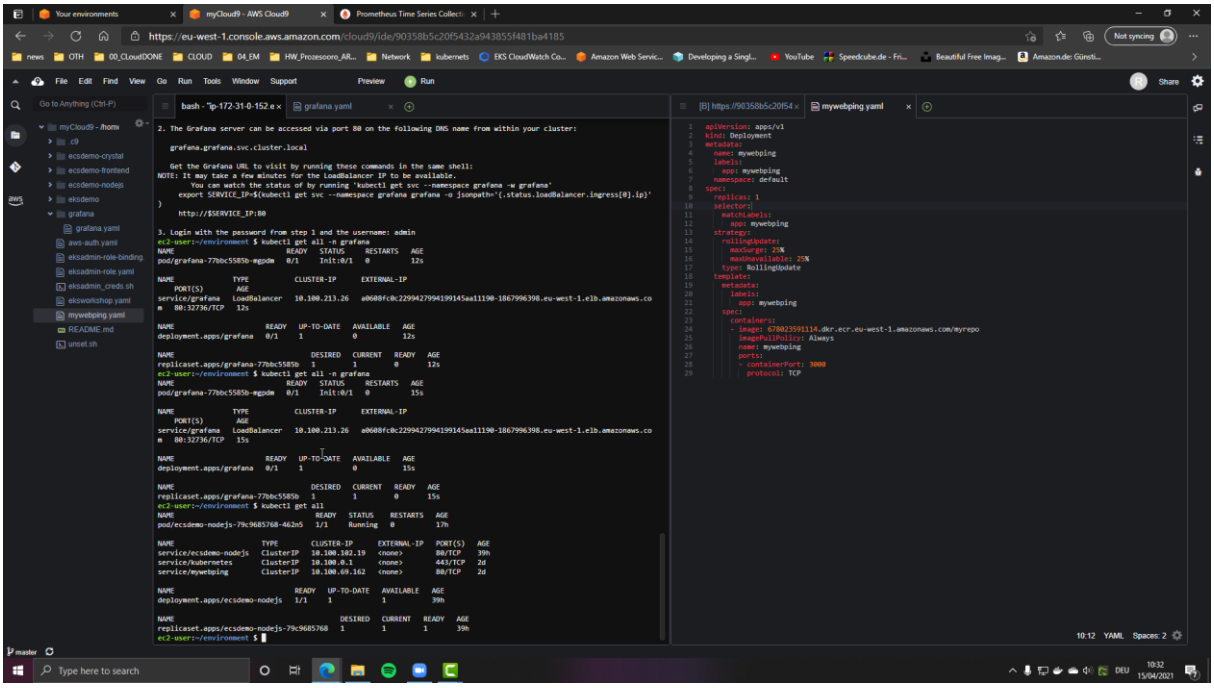
$kubectl apply -f <name_of_file>

For example see 02_yaml_json\04_nodejs_backend_API_deployment.yaml and 02_yaml_json\04_nodejs_backend_API_service.yaml. Further microservies like described on this site https://www.eksworkshop.com/beginner/050_deploy/ were deployed to get some workload on the cluster and I experienced with different functionalities EKS offers like e.g. Elastic Load Balancer.

But as getting really interested about Containers and Container orchestration I experienced with Docker and used the AWS service ECR to deploy a container on my Kubernetes Cluster.
Thereafter I deployed an container image to the Amazon Elastic Container registry and applied a simple yaml file to deploy my app called “mywebping.yaml” in the Cloud9 environment on the terminal. This implies in a pod running on my cluster.
The source code of the app can be inspected in the folder 01_code\mywebping. This source code is available open source and originates from the book “Learn Docker in a Month of Lunches” by Elton

Stoneman. (see github repo: [diamol/ch03/exercises/web-ping-optimized at master · sixeyed/diamol (github.com)](https://github.com/sixeyed/diamol/tree/master/ch03/exercises/web-ping-optimized))

>>>also see 02_yaml_json\mywebping.yaml

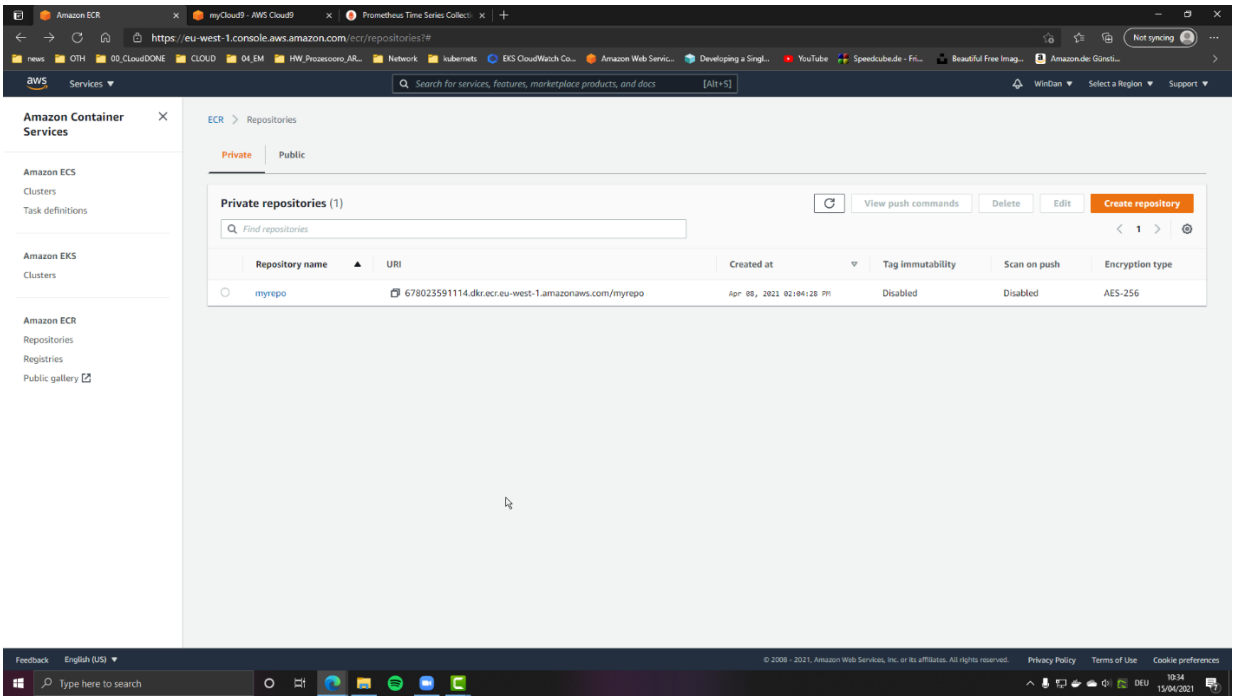


Screenshot 3: Cloud9 mywebping

Furtheron I experienced with topics like autoscaling my applications and clusters like described on this site: https://www.eksworkshop.com/beginner/080_scaling/
In particularly I focused on configuring the Horizontal Pod Autoscaler on a specific app, on which I created Load to see how the autoscaling works.
As another little practice I assigned pods to nodes using a nodeSelector specified on the node and in the yaml file, with which I deployed a sample application

Unfortunately, as those two topics was only a side quest I have no prove about successfully implementing this.

But as I really spend many hours on this topic I also came in touch with autoscaling groups and Instance refreshment. (This refers to section 3.3) This part is proved in the first video from minute 3:30 onwards, see >>>03_videos\01_Cloud9.mkv



Screenshot 4: ECR

3.5 Role Based Access Control (RBAC)

As Security is still an uprising topic and becomes more and more important for companies, especially if those companies are using the Cloud for compute power or as storage unit for development purposes.
First, I want to mention two important terms and will explain the core difference of them:
Authentication & Authorization
Authentication is, that you identify yourself whereas Authorization proves if you are allowed to enter a specific area, which may be virtual or also could be in the real world.

What is RBAC? Informal RBAC can be seen as extra Identity and Access management for in-cluster kubernetes resources, with all the policies, roles, group concepts, but slightly different to AWS IAM. RBAC is the kubernetes native access control and provides above-described functionality. RBAC is important if no other access control system is available and as Kubernetes is built to run in every environment kubernetes RBAC is important and therefore kubernetes does not only depend on other Identity and Access Management Services.

For the Authorization part for Kubernetes there are two different kind of roles and role bindings: role vs. clusterrole and rolebinding vs. clusterrole-binding.
In my case I used the clusterrole to enable a user to view alle resources and details on “myEKS” cluster. The clusterrole offers a cluster wide access and in contrary to “normal” roles is not restricted to a particular namespace. For accomplishing this I created a user and additionally applied the clusterrole and the clusterrole-binding to the user. This two yaml file I applied can be viewed in the folder
>>> 02_yaml_json\05_rbac_eksadmin_role.yaml and 02_yaml_json\05_rbac_eksadmin_role-binding.yaml

The step for step guide for using RBAC is available here:
https://www.eksworkshop.com/beginner/090_rbac/

And the Kubernetes documentation for using RBAC authorization is accessible via this link: https://kubernetes.io/docs/reference/access-authn-authz/rbac/

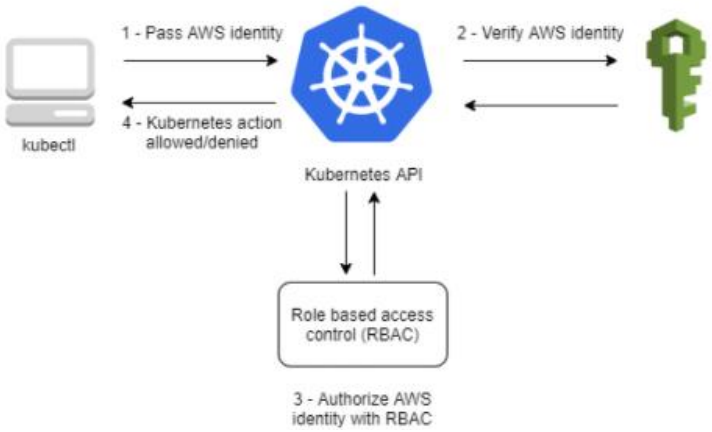


Image 1: Cluster Authentication

The image above shows the coherences of AWS IAM and kubernetes RBAC.

3.6 Monitoring

For monitoring purposes, I used three different software tools.

As first and easiest one to set up I deployed the official **Kubernetes Dashboard** by entering in the Cloud9 Terminal:

```
$kubectl apply -f \
https://raw.githubusercontent.com/kubernetes/dashboard/v2.0.0/aio/deploy/recommended.yaml
```

As the deployment is finished, we need to access it via a proxy because it is only deployed in my private cluster. The kube-proxy command forwards our requests to the dashboard service:

```
$kubectl proxy –port=8080 –address=0.0.0.0 –disable-filter=true &
```

This results in starting the proxy and listening on port 8080. The flag "--address=0.0.0.0" enables listening on all interfaces and the flag "—disable-filter=true" disables filtering of non-localhost requests. The "&" effects, that this command will continue to run in the background of the current terminal's session. To view the kubernetes dashboard there were two main different options: first was to use the preview (Tools / Preview / Preview Running Application) feature of the Cloud9 environment and appending "/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy" to the end of the URL. Second option was to view the kubernetes dashboard via the URL of the hosting Cloud9 environment with above mentioned string appended: "https://90358b5c20f5432a943855f481ba4185.vfs.cloud9.eu-west-1.amazonaws.com//api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy"

(view video >>>03_videos\02_dashboards.mkv)

(view step for step guide at: https://www.eksworkshop.com/beginner/040_dashboard/)

For following monitoring tools helm is required: “Helm is a package manager for Kubernetes that packages multiple Kubernetes resources into a single logical deployment unit called a Chart. Charts are easy to create, version, share, and publish.”
(https://www.eksworkshop.com/beginner/060_helm/)

As second monitoring system I deployed **Prometheus**. (also for log aggregation)
Prometheus is an open-source system monitoring and alerting toolkit originally built at SoundCloud.
The Prometheus helm repo was added by the command:

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

For deploying Prometheus an additional namespace “prometheus” was created.

```
$kubectl create namespace prometheus
```

Gp2 EBS volumes for simplicity and the standard configuration is used.

```
$helm install prometheus prometheus-community/prometheus \
  --namespace prometheus \
  --set alertmanager.persistentVolume.storageClass="gp2" \
  --set server.persistentVolume.storageClass="gp2"
```

After checking if Prometheus components are deployed as expected …

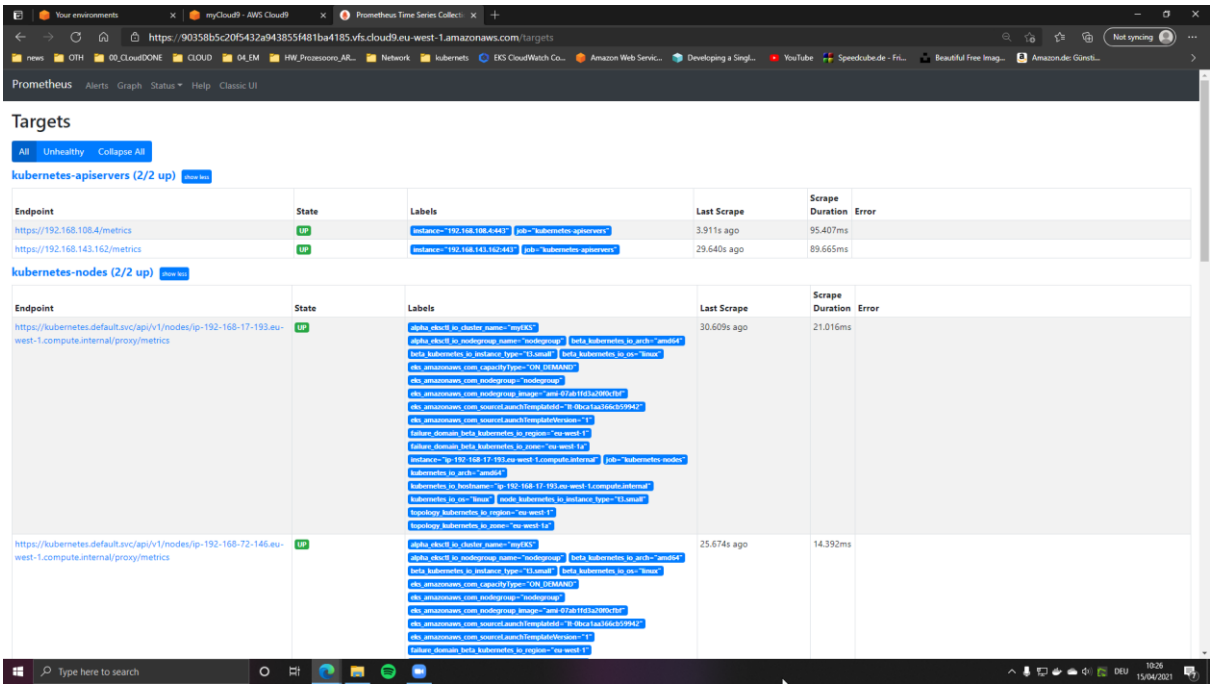
```
$kubectl get all -n Prometheus
```

… and all pods are running I continued reaching Prometheus.

To access Prometheus, the kubectl port-forward command is used to reach the application.
In Cloud9 Terminal:

```
$kubectl port-forward -n Prometheus deploy/Prometheus-server 8080:9090
```

Same as by the Kubernetes Dashboard Prometheus can be viewed in the preview feature of Cloud 9 (Tools / Preview / Preview Running Application) by append the string “/targets” to the end of the URL.



Screenshot 5: Prometheus

As third monitoring system I deployed Grafana.
Grafana is an open source visualization and analytics software. It allows querying, visualizing, alerting on and exploring my metrics, e.g. those from Prometheus.
In the following several parameters are overridden but primarily Grafana defaults are used. Storage class is set to gp2 like Prometheus. The data source is set to point to Prometheus and an external load balancer is created for the service.

A yaml file was created. See >>>02_yaml_json\07_grafana.yaml

After creating the namespace “grafana” Grafana was deployed with following commands:

```
$kubectl create namespace grafana

$helm install grafana grafana/grafana \
    --namespace grafana \
    --set persistence.storageClassName="gp2" \
    --set persistence.enabled=true \
    --set adminPassword='EKS!sAWSome' \
    --values ${HOME}/environment/grafana/grafana.yaml \
    --set service.type=LoadBalancer
```

After checking that Grafana is deployed properly…

```
$kubectl get all -n Grafana
```

… I exported the Grafana ELB URL using following commands.

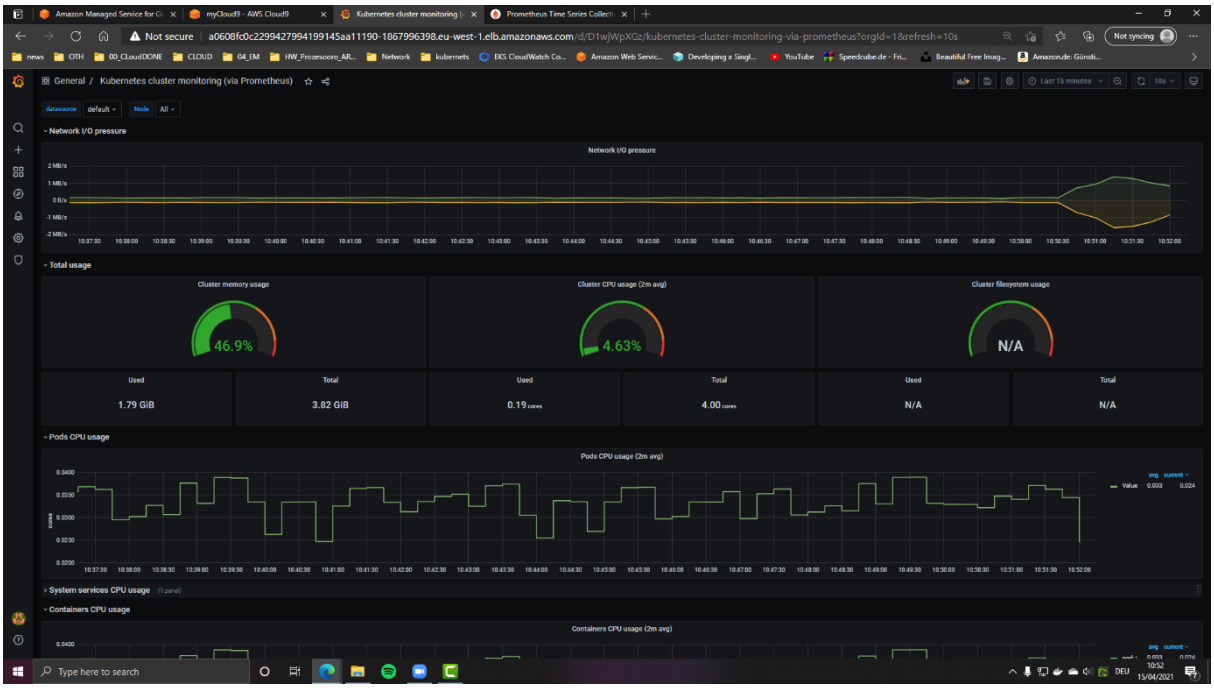
```
$export ELB=$(kubectl get svc -n grafana grafana -o \
  jsonpath='{.status.loadBalancer.ingress[0].hostname}')

$echo http://$ELB
```

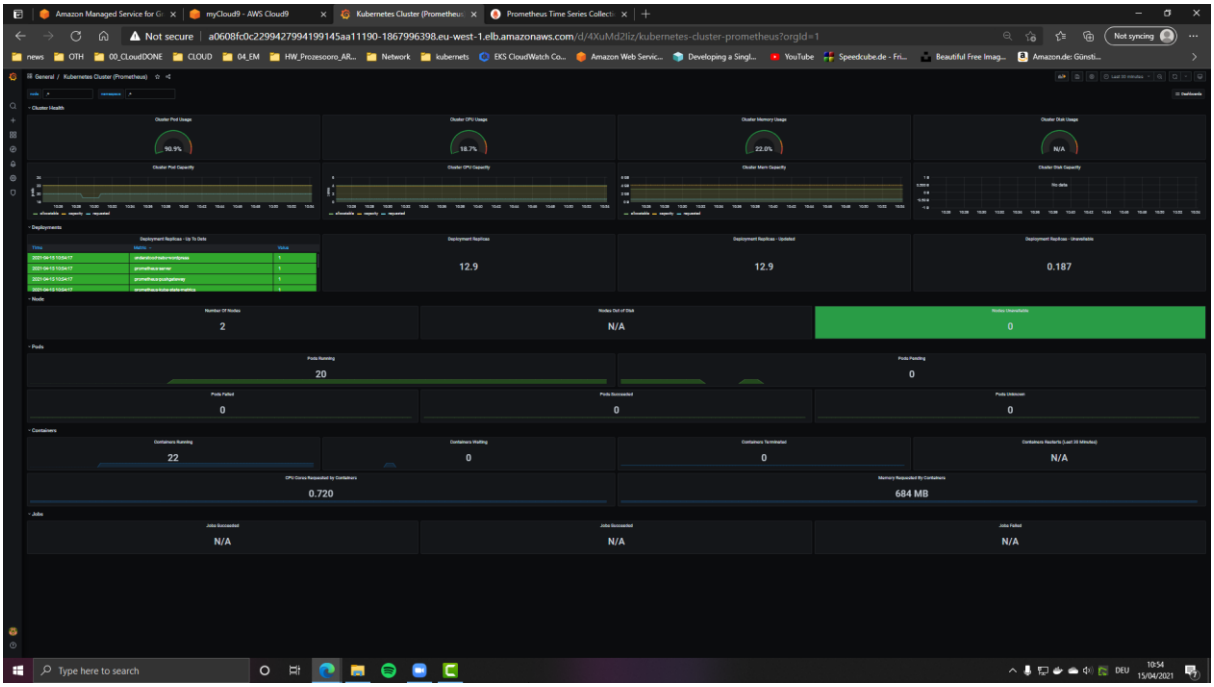
For Login, username *admin* and password *EKS!sAWSome* are used.
Password can bet get by running following command:

```
$ kubectl get secret --namespace grafana grafana -o jsonpath="{.data.admin-password}" | \
  base64 --decode ; echo
```

After Login I have created two different Dashboards: Cluster Monitoring Dashboard and Pods Monitoring Dashboard. (for creation steps see: https://www.eksworkshop.com/intermediate/240_monitoring/dashboards/)



Screenshot 6: Grafana Cluster Monitoring



Screenshot 7: Grafana Pods Monitoring

Cleanup commands:

###Kubernetes Dashboard

kill proxy

$pkill -f 'kubectl proxy --port=8080'

delete dashboard

$kubectl delete -f \

https://raw.githubusercontent.com/kubernetes/dashboard/v2.0.0/aio/deploy/recommended.yaml

###Prometheus

$unset DASHBOARD_VERSION

$helm uninstall prometheus --namespace prometheus

$kubectl delete ns prometheus

###Grafana

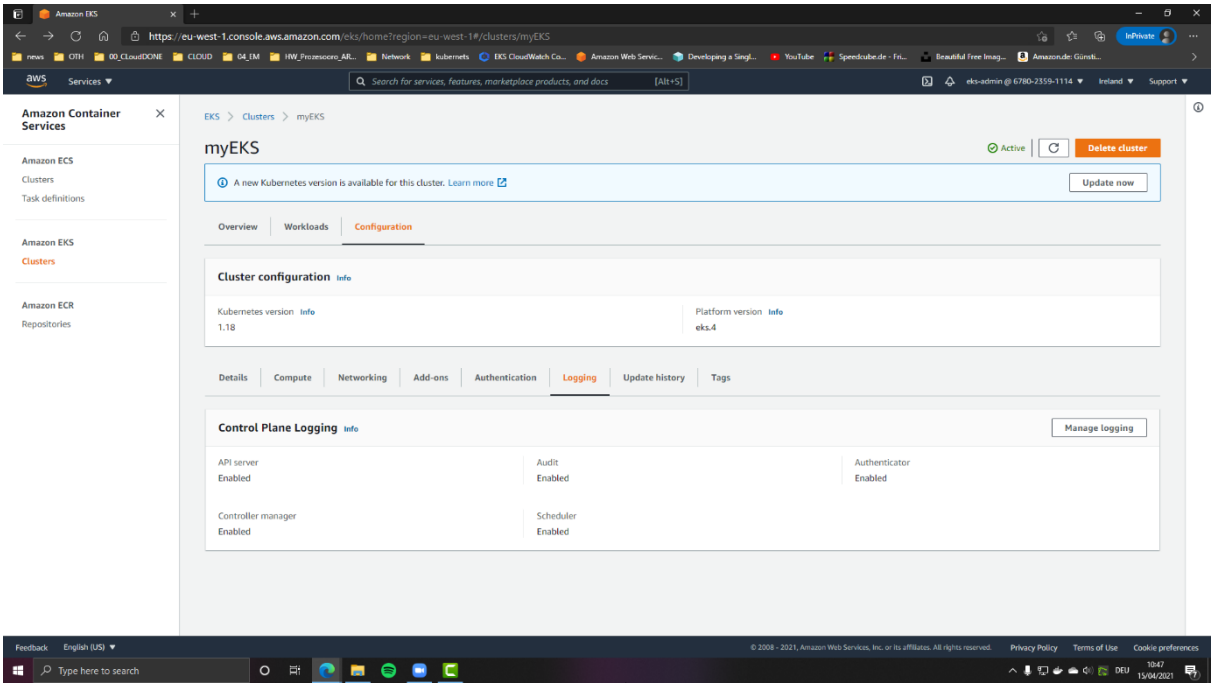
$helm uninstall grafana --namespace grafana

$kubectl delete ns grafana

$rm -rf ${HOME}/environment/grafana

3.7 Logging with CloudWatch (using 2.6 CloudWatch)

Logging of “myEKS” Cluster was uncomplicated to set up. Therefore, only Logging options like API-Server, Audit, Authenticator, Controller Manager and Scheduler had to be enabled under Compute > Logging. (see Screenshot below)



Screenshot 8: myEKS Logging

As second Logging option I chose to utilize the new CloudWatch Container Insights to see how native CloudWatch features can be used to monitor my EKS Cluster performance. I use CloudWatch Container Insights especially for collecting and aggregating logs from my containerized applications, in particularly the running container “mywebping”. Also CloudWatch Container insights can

summarize metrics and logs from my containerized applications. So, for example these metrics include diagnostic information about logs of “mywebping” (log events).

As first step I installed the CloudWatch Agent to my EKS cluster. For this the appropriated policy “CloudWatchAgentServerPolicy” has to be attached to the role used for installing Container Insights. Then only following command has to be run in my Cloud9 Terminal:

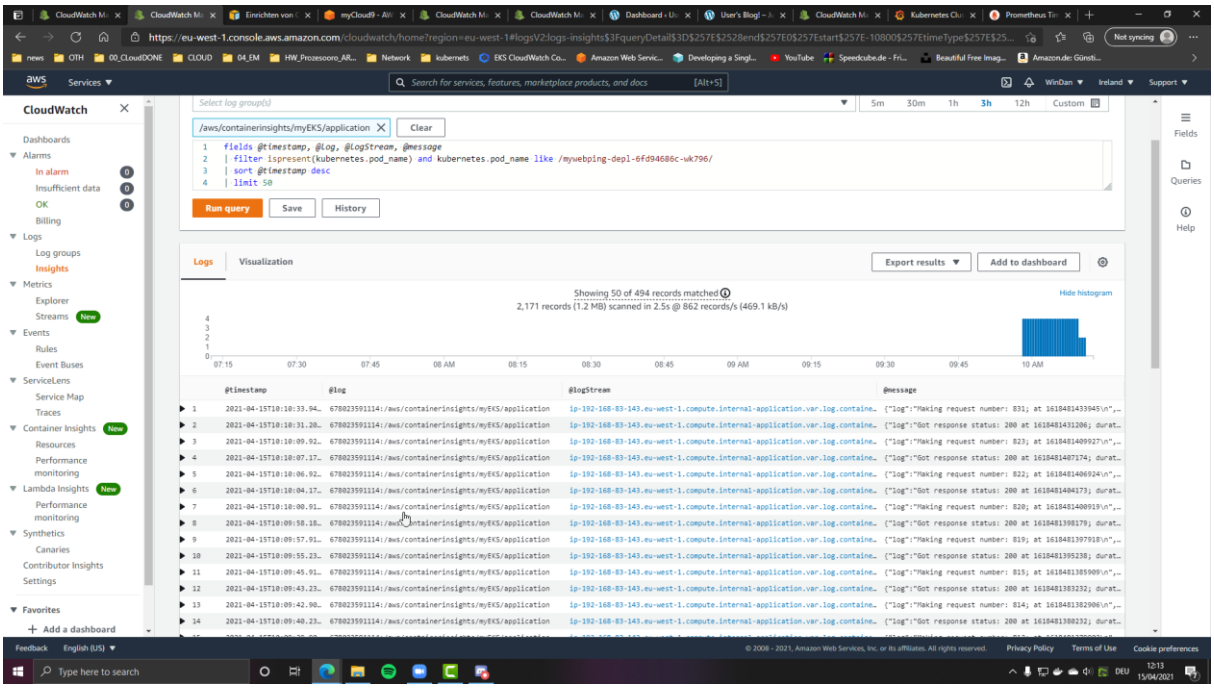
> $curl -s https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-insights-monitoring/quickstart/cwagent-fluentd-quickstart.yaml | sed "s/myEKS/Cloud9_eksworkshop/;s/eu-west-1/eu-west-1/" | kubectl apply -f –

This command above will create a namespaces “amazon-cloudwatch”, then creates all necessary security object for both DaemonSet (including SecurityAccount, ClusterRole, ClusterBinding), then deploys the CloudWatch-Agent (responsible for sending the metrics to CLoudWatch) and Fluentd (responsible for sending logs to Cloudwatch) both as a DaemonSet. Finally the ConfigMap configurations for both DameonSets are deployed.
For verifying that the data is being collected in CloudWatch, I had to launch the CloudWatch Containers UI in my browser using following link:

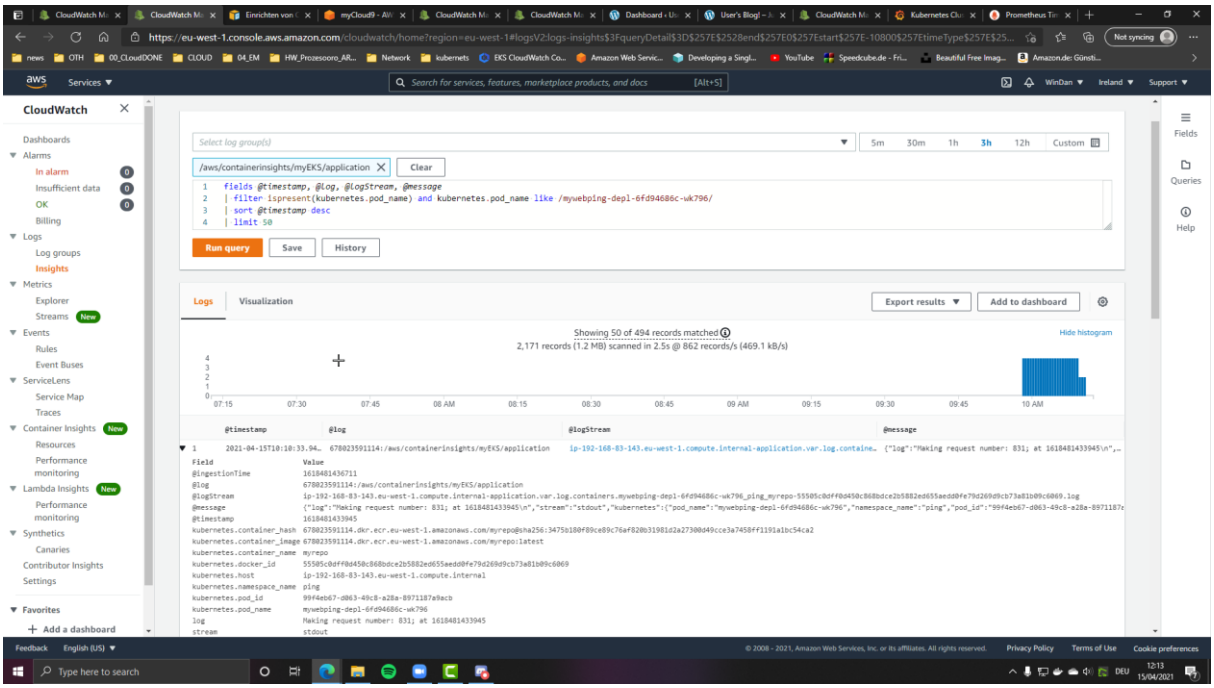
> https://console.aws.amazon.com/cloudwatch/home?region=eu-west-1#cw:dashboard=Container;context=~(clusters~'eksworkshopeksctl~dimensions~(~)~performanceType~'Service)

(see: https://www.eksworkshop.com/intermediate/250_cloudwatch_container_insights/ and https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/Container-Insights-setup-EKS-quickstart.html)

The following three Screenshots will give an imagination how these looks like:



Screenshot 9: Logging Container Insights1



Screenshot 10: Logging Container Insights2

4. Summary & Outlook

4.1 Summary

This Lab exercise was a big challenge for me as I had to learn Docker and Kubernetes in its core concept and how all this is playing together. Later on, I had real troubles creating a EKS cluster because of different problems concerning Identity and Access Management in AWS. Also I struggled with getting started as there were so many documentations and tutorials and by following along a few of them I failed because of my lack of knowledge as newbie in terms of Cloud Computing. Therefore the www.eksworkshop.com tutorial helped me out and was almost for all parts the one source besides Kubernetes topics regarding RBAC. So, one thing is left so say: I really enjoyed getting into this topic but also it was frustrating at least sometimes as I had real difficulties to follow along some tutorials. All in all it was very challenging but definitely worth spending my time because I learned so many different skills and I got a better imagination what the actual state of Cloud Computing is but also how modern applications are deployed.

4.2 Outlook

There is always room for improvement and therefore I want to mention some aspects where I recognized that it could make sense to change and improve things. For example, instead of running Prometheus and Grafana as pods on my cluster there is perhaps a better alternative, Amazon offers two AWS services for those: Amazon Managed Service for Grafana and Amazon Managed Service for Prometheus. Using these two Services it would cost less time managing Prometheus and Grafana infrastructures and more time would be left for managing other applications. Another improvement could be to switch from the manage node group based on EC2 instances to the serverless compute engine for containers AWS Fargate. AWS Fargate eliminates the need to provision and manage node groups and only produces costs for resources needed by the running applications. Also it would improve security through its applications isolation by design. And one last topic I want to mention: There are still lot more possibilities if it comes to Logging and Monitoring where other approaches with the ELK stack could be taken but also additional AWS Services like AWS X-Ray or AWS CloudTrail could be used.

5. Sources

5.1 List of Figures

Images:

Image 1: Cluster Authentication - https://docs.aws.amazon.com/eks/latest/userguide/managing-auth.html

Screenshots:

Screenshot 1: Architectural Overview - >>> 04_screenshots\ArchitecturalOverview.png

Screenshot 2: Cloud9 environment - >>> 04_screenshots\01_Cloud9\Cloud9_all_pods_services.png

Screenshot 3: Cloud9 mywebping - >>> 04_screenshots\01_Cloud9\cloud9_example.png

Screenshot 4: ECR - >>> 04_screenshots\03_ECR\ECR.png

Screenshot 5: Prometheus - >>> 04_screenshots\04_Monitoring\prometheus.png

Screenshot 6: Grafana Cluster Monitoring - >>> 04_screenshots\04_Monitoring\grafana_cluster_01.png

Screenshot 7: Grafana Pods Monitoring - >>> 04_screenshots\04_Monitoring\grafana_pods_01.png

Screenshot 8: myEKS Logging - >>> 04_screenshots\05_CloudWatch\myEKS_loggingCluster.png

Screenshot 9: Logging Container Insights1 - >>> 04_screenshots\05_CloudWatch\CloudWatch_logs.png

Screenshot 10: Logging Container Insights2 - >>> 04_screenshots\05_CloudWatch\CloudWatch_logs.png

5.2 Sources

Internet sources:

Amazon Logos: https://aws.amazon.com/architecture/icons/, last visited on 27.04.2021

Amazon EKS workshop: https://www.eksworkshop.com, last visited on 27.04.2021

Amazon EKS documentation: https://docs.aws.amazon.com/eks/latest/userguide/getting-started.html, last visited on 27.04.2021

Kubernetes: https://kubernetes.io/docs/tasks/access-application-cluster/web-ui-dashboard/, last visited on 27.04.2021

Kubernetes RBAC: https://kubernetes.io/docs/reference/access-authn-authz/rbac/, last visited on 27.04.2021

Books:

Learn Docker in a Month of Lunches – Elton Stoneman, publisher: Manning

Learn Kubernetes in a Month of Lunches – Elton Stoneman, publisher: Manning