National College of Ireland

Configuration Manual

Internship
MSc in Cybersecurity

Louise Elie
Student ID: 21195137

School of Computing
National College of Ireland

Supervisor: Kamil Mahajan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Louise Elie
Student ID:	21195137
Programme:	MSc in Cybersecurity **Year:** 2023-2024
Module:	Internship
Lecturer:	Kamil Mahajan
Submission Due Date:	September 16th, 2024
Project Title:	Optimising Digital Forensics Investigations in Containers as a Service Environments
Word Count:	2717 words **Page Count:** 24 pages

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: 

Date: September 13th, 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Louise Elie
Student ID: 21195137

1 Introduction

The lab environment created as part of the research project is described in this manual, along with additional technical information and guidance on the configuration. A containerised WordPress application was hosted on the managed service Azure Kubernetes Service to conduct digital forensics and gather data from containers runtime events and metadata. This document includes details about system configuration (hardware and software), infrastructure and application configuration, activities performed on attacker's side, data collection, and hardening solutions. The main paper on the research provides insights into research methodology, design specification, and implementation.

2 System configuration

2.1 Hardware configuration

Table 1 contains the details of the hardware components and the operating systems for the two machines used in the tests: the first one deploys the infrastructure on Azure portal and Azure Cloud Shell, while the second one is the virtual machine to simulate attacks.

Table 1: Hardware and system specifications

Profile	CPU	RAM	Disk Space	Operating system
Client machine	Intel(R) Core(TM) i7-6500U CPU @ 2.50GHz 2.59 GHz	8.00 GB	ADATA SX8200PNP SSD 1TB	Windows 11 Pro 22H2
Attacker machine	2 CPU	2.00 GB	20 GB	Kali Linux 2024.2

2.2 Software configuration

Table 2 lists the different software tools that were part of the lab environment, and their details such as a brief description and the version used.

Table 2: Software specifications

Tool	Description	Version
Azure Cloud Shell[1]	Authenticated and interactive terminal to manage Azure resources, with Bash opted as the shell language.	2.62.0
Kubernetes[2]	Container orchestration tool managed by Azure under the name of Azure Kubernetes Services (AKS).	1.28.9
Helm[3]	Kubernetes packet manager used to deploy an ingress controller and WordPress.	3.14.2
WordPress[4]	Content Management System (CMS) for admin-friendly websites creation.	6.5.3
MySQL[5]	Database management system utilized to store WordPress content and managed by Azure Database for MySQL.	8.0.21
X Brute Forcer[6]	Open-source and simple brute-force script.	1.2
Apache JMeter[7]	Open-source tool for load testing used to launch DoS attacks.	5.6.3
WPS Limit Login[8]	WordPress add-on that hardens login attempts to the web application.	1.5.9.1

3 Lab implementation guideline

This step-by-step guide provides instructions performed to replicate the deployment of the lab environment. Azure Cloud Shell was used to configure the different components as described in this tutorial (*Microsoft Learn*, 2024) and to access data from container runtime events and metadata. Detailed commands and configuration files are available in the ICT solution folder.

3.1 Prerequisites

The research requires the following elements to ensure the proper functioning of the attack simulations on the hosted containerised web application:

- Create an Azure free account.
- Configure Azure Cloud Shell on your machine and select Bash as the preferred shell.
- Deploy Helm on your Azure Cloud Shell, and Bitnami Helm repository.
- Create a Kali Linux virtual machine on a hypervisor (e.g. VirtualBox).
- Download and install OpenJDK (8+ version) for Apache JMeter on Kali Linux.
- On Azure Cloud Shell, define the environment variables as shown in Figure 1.

[1] https://azure.microsoft.com/en-us/get-started/azure-portal/cloud-shell
[2] https://learn.microsoft.com/en-us/azure/aks/
[3] https://github.com/helm/helm
[4] https://wordpress.org/documentation/wordpress-version/version-6-5-3/
[5] https://learn.microsoft.com/en-us/azure/mysql/
[6] https://github.com/bibortone/XBruteForcer
[7] https://github.com/apache/jmeter
[8] https://wordpress.org/plugins/wps-limit-login/

```
export SSL_EMAIL_ADDRESS="$(az account show --query user.name --output tsv)"
export NETWORK_PREFIX="$(($RANDOM % 253 + 1))"
export RANDOM_ID="$(openssl rand -hex 3)"
export MY_RESOURCE_GROUP_NAME="myWordPressAKSResourceGroup$RANDOM_ID"
export REGION="northeurope"
export MY_AKS_CLUSTER_NAME="myAKSCluster$RANDOM_ID"
export MY_PUBLIC_IP_NAME="myPublicIP$RANDOM_ID"
export MY_DNS_LABEL="mydnslabel$RANDOM_ID"
export MY_VNET_NAME="myVNet$RANDOM_ID"
export MY_VNET_PREFIX="10.$NETWORK_PREFIX.0.0/16"
export MY_SN_NAME="mySN$RANDOM_ID"
export MY_SN_PREFIX="10.$NETWORK_PREFIX.0.0/22"
export MY_MYSQL_DB_NAME="mydb$RANDOM_ID"
export MY_MYSQL_ADMIN_USERNAME="dbadmin$RANDOM_ID"
export MY_MYSQL_ADMIN_PW="$(openssl rand -base64 32)"
export MY_MYSQL_SN_NAME="myMySQLSN$RANDOM_ID"
export MY_MYSQL_HOSTNAME="$MY_MYSQL_DB_NAME.mysql.database.azure.com"
export MY_WP_ADMIN_PW="$(openssl rand -base64 32)"
export MY_WP_ADMIN_USER="wpcliadmin"
export FQDN="${MY_DNS_LABEL}.${REGION}.cloudapp.azure
```

Figure 1: Environment variables definition

3.2 Azure resources

The lab environment is hosted on Azure, a public cloud. The project needs two essential components for the cloud infrastructure: a resource group, and a virtual network with a subnet. As presented in Figure 2, a resource group must be created first with the `az group create` command. It is a logical group containing Azure resources.

```
louise [ ~ ]$ az group create \
    --name $MY_RESOURCE_GROUP_NAME \
    --location $REGION
{
  "id": "/subscriptions/c66088df-0902-4594-8569-aef412115d70/resourceGroups/myWordPressAKSResourceGroupbc8a01",
  "location": "northeurope",
  "managedBy": null,
  "name": "myWordPressAKSResourceGroupbc8a01",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null,
  "type": "Microsoft.Resources/resourceGroups"
}
```

Figure 2: Azure resource group creation

Then, a virtual network must be created with an associated subnet with the `az network vnet create` command. Figure 3 displays the command output. The virtual network enables Kubernetes resources to securely communicate.

```
louise [ ~ ]$ az network vnet create \
    --resource-group $MY_RESOURCE_GROUP_NAME \
    --location $REGION \
    --name $MY_VNET_NAME \
    --address-prefix $MY_VNET_PREFIX \
    --subnet-name $MY_SN_NAME \
    --subnet-prefixes $MY_SN_PREFIX
{
  "newVNet": {
    "addressSpace": {
      "addressPrefixes": [
        "10.177.0.0/16"
      ]
    },
    "enableDdosProtection": false,
    "etag": "W/\"ef14376f-861f-41d6-a545-2fdc43971638\"",
    "id": "/subscriptions/c66088df-0902-4594-8569-aef412115d70/resourceGroups/myWordPressAKSResourceGroupbc8a01/providers/Microsoft
.Network/virtualNetworks/myVNetbc8a01",
    "location": "northeurope",
    "name": "myVNetbc8a01",
    "provisioningState": "Succeeded",
    "resourceGroup": "myWordPressAKSResourceGroupbc8a01",
    "resourceGuid": "c4d1b271-12b9-43db-8bd4-ef2add666fd9",
    "subnets": [
      {
        "addressPrefix": "10.177.0.0/22",
        "delegations": [],
        "etag": "W/\"ef14376f-861f-41d6-a545-2fdc43971638\"",
        "id": "/subscriptions/c66088df-0902-4594-8569-aef412115d70/resourceGroups/myWordPressAKSResourceGroupbc8a01/providers/Micro
soft.Network/virtualNetworks/myVNetbc8a01/subnets/mySNbc8a01",
        "name": "mySNbc8a01",
        "privateEndpointNetworkPolicies": "Disabled",
        "privateLinkServiceNetworkPolicies": "Enabled",
        "provisioningState": "Succeeded",
        "resourceGroup": "myWordPressAKSResourceGroupbc8a01",
        "type": "Microsoft.Network/virtualNetworks/subnets"
      }
    ],
    "type": "Microsoft.Network/virtualNetworks",
    "virtualNetworkPeerings": []
  }
}
```

Figure 3: Azure virtual network creation

3.3 Azure database

The Azure Database for MySQL stores WordPress media content. In Figure 4, among its attributes, the `az mysql flexible-server create` command indicates the server's admin username and password, the compute tier (Burstable), the compute size (Standard_B2s), the retention period of seven days, and the MySQL version (specified in the previous section).

```
louise [ ~ ]$ az mysql flexible-server create \
    --admin-password $MY_MYSQL_ADMIN_PW \
    --admin-user $MY_MYSQL_ADMIN_USERNAME \
    --auto-scale-iops Disabled \
    --high-availability Disabled \
    --iops 500 \
    --location $REGION \
    --name $MY_MYSQL_DB_NAME \
    --database-name wordpress \
    --resource-group $MY_RESOURCE_GROUP_NAME \
    --sku-name Standard_B2s \
    --storage-auto-grow Disabled \
    --storage-size 20 \
    --subnet $MY_MYSQL_SN_NAME \
    --private-dns-zone $MY_DNS_LABEL.private.mysql.database.azure.com \
    --tier Burstable \
    --version 8.0.21 \
    --vnet $MY_VNET_NAME \
    --yes -o JSON
Checking the existence of the resource group 'myWordPressAKSResourceGroupbc8a01'...
Resource group 'myWordPressAKSResourceGroupbc8a01' exists ? : True
You have supplied a Vnet and Subnet name. Verifying its existence...
Using existing Vnet "myVNetbc8a01" in resource group "myWordPressAKSResourceGroupbc8a01"
The address prefix does not exist in the Vnet. Adding address prefix 10.0.0.0/16 to Vnet myVNetbc8a01.
Creating new Subnet "myMySQLSNbc8a01" in resource group "myWordPressAKSResourceGroupbc8a01"
Creating a private dns zone mydnslabelbc8a01.private.mysql.database.azure.com in resource group "myWordPressAKSResourceGroupbc8a01"
IOPS is 500 which is either your input or free(maximum) IOPS supported for your storage size and SKU.
Creating MySQL Server 'mydbbc8a01' in group 'myWordPressAKSResourceGroupbc8a01'...
Your server 'mydbbc8a01' is using sku 'Standard_B2s' (Paid Tier). Please refer to https://aka.ms/mysql-pricing for pricing details
Creating MySQL database 'wordpress'...
Make a note of your password. If you forget, you would have to reset your password with'az mysql flexible-server update -n mydbbc8a
01 -g myWordPressAKSResourceGroupbc8a01 -p <new-password>'.
Try using az 'mysql flexible-server connect' command to test out connection.
```

Figure 4: Azure Database for MySQL creation

Disable SSL connection with the `az mysql flexible-server parameter set` command. in Figure 5, to include WordPress integration.

```
louise [ ~ ]$ az mysql flexible-server parameter set \
    -g $MY_RESOURCE_GROUP_NAME \
    -s $MY_MYSQL_DB_NAME \
    -n require_secure_transport -v "OFF" -o JSON
Readonly attribute name will be ignored in class <class 'azure.mgmt.rdbms.mysql_flexibleservers.models._models_py3.Configuration'>
```

Figure 5: Disabling SSL connection

3.4 Azure Kubernetes Service

Once the core resources and the database server are deployed on Azure, the AKS cluster can be created. Figure 6 presents the `az aks create` command used with its parameters. By default, AKS will select 1.28.9 as Kubernetes's default version.

```
louise [ ~ ]$ export MY_SN_ID=$(az network vnet subnet list --resource-group $MY_RESOURCE_GROUP_NAME --vnet-name $MY_VNET_NAME --qu
ery "[0].id" --output tsv)
```

```
louise [ ~ ]$
az aks create \
    --resource-group $MY_RESOURCE_GROUP_NAME \
    --name $MY_AKS_CLUSTER_NAME \
    --auto-upgrade-channel stable \
    --enable-cluster-autoscaler \
    --enable-addons monitoring \
    --location $REGION \
    --node-count 1 \
    --min-count 1 \
    --max-count 3 \
    --network-plugin azure \
    --network-policy azure \
    --vnet-subnet-id $MY_SN_ID \
    --no-ssh-key \
    --node-vm-size Standard_DS2_v2 \
    --service-cidr 10.255.0.0/24 \
    --dns-service-ip 10.255.0.10 \
    --zones 1 2 3
docker_bridge_cidr is not a known attribute of class <class 'azure.mgmt.containerservice.v2024_02_01.models._models_py3.ContainerSe
rviceNetworkProfile'> and will be ignored
```

Figure 6: AKS cluster creation

Next actions will be performed on the newly created cluster. To connect to the Kubernetes cluster, the kubectl command-line client is used. It is already installed in Azure Cloud Shell. Figure 7 displays the `az aks get-credentials` command which role is to download credentials to use kubectl.

```
louise [ ~ ]$ az aks get-credentials --resource-group $MY_RESOURCE_GROUP_NAME --name $MY_AKS_CLUSTER_NAME --overwrite-existing
Merged "myAKSCluster86029c" as current context in /home/louise/.kube/config
```

Figure 7: kubectl configuration

As presented in Figure 8, connection is successful as cluster nodes are listed with the `kubectl get nodes` command.

```
louise [ ~ ]$ kubectl get nodes
NAME                                STATUS   ROLES   AGE   VERSION
aks-nodepool1-35586775-vmss000000   Ready    agent   12m   v1.28.9
```

Figure 8: List of cluster nodes

To expose the web application, an ingress controller must be created. It has the role of both a load balancer and a reverse proxy. The ingress controller is configured with a static public IP address. Figure 9 presents the command to install the ingress-nginx add-on via Helm. First, the local Helm Chart repository cache must be updated and the ingress-nginx Helm repository added.

```
louise [ ~ ]$ export MY_STATIC_IP=$(az network public-ip create --resource-group MC_${MY_RESOURCE_GROUP_NAME}_${MY_AKS_CLUSTER_NAME
}_${REGION} --location ${REGION} --name ${MY_PUBLIC_IP_NAME} --dns-name ${MY_DNS_LABEL} --sku Standard --allocation-method static -
-version IPv4 --zone 1 2 3 --query publicIp.ipAddress -o tsv)
louise [ ~ ]$ helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
    helm repo update
    helm upgrade --install --cleanup-on-fail --atomic ingress-nginx ingress-nginx/ingress-nginx \
        --namespace ingress-nginx \
        --create-namespace \
        --set controller.service.annotations."service\.beta\.kubernetes\.io/azure-dns-label-name"=$MY_DNS_LABEL \
        --set controller.service.loadBalancerIP=$MY_STATIC_IP \
        --set controller.service.annotations."service\.beta\.kubernetes\.io/azure-load-balancer-health-probe-request-path"=/healthz
 \
        --wait --timeout 10m0s
"ingress-nginx" has been added to your repositories
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "ingress-nginx" chart repository
Update Complete. ⎈Happy Helming!⎈
Release "ingress-nginx" does not exist. Installing it now.
NAME: ingress-nginx
LAST DEPLOYED: Tue Jun 18 09:12:27 2024
NAMESPACE: ingress-nginx
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
The ingress-nginx controller has been installed.
It may take a few minutes for the load balancer IP to be available.
You can watch the status by running 'kubectl get service --namespace ingress-nginx ingress-nginx-controller --output wide --watch'
```

Figure 9: Ingress controller deployment

3.5 WordPress application

WordPress is installed via the Helm chart built by Bitnami (see prerequisites). It also uses a local MariaDB as the database, which needs to be replaced by Azure Database for MySQL. Figure 10 displays the installation command.

```
louise [ ~/practikube ]$ helm upgrade --install --cleanup-on-fail \
    --wait --timeout 10m0s \
    --namespace wordpress \
    --create-namespace \
    --set wordpressUsername="$MY_WP_ADMIN_USER" \
    --set wordpressPassword="$MY_WP_ADMIN_PW" \
    --set wordpressEmail="$SSL_EMAIL_ADDRESS" \
    --set externalDatabase.host="$MY_MYSQL_HOSTNAME" \
    --set externalDatabase.user="$MY_MYSQL_ADMIN_USERNAME" \
    --set externalDatabase.password="$MY_MYSQL_ADMIN_PW" \
    --set ingress.hostname="$FQDN" \
    wordpress bitnami/wordpress
Release "wordpress" does not exist. Installing it now.
```

Figure 10: WordPress installation via Helm

As presented in Figure 11 and confirmed in Figure 12, once WordPress is installed, the web application's IP address is available by fetching the Kubernetes service IP. The command also indicates information to obtain the credentials provided during the installation.

```
Release "wordpress" does not exist. Installing it now.
NAME: wordpress
LAST DEPLOYED: Thu Jun 20 10:20:15 2024
NAMESPACE: wordpress
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
CHART NAME: wordpress
CHART VERSION: 22.4.13
APP VERSION: 6.5.4

** Please be patient while the chart is being deployed **

Your WordPress site can be accessed through the following DNS name from within your cluster:

    wordpress.wordpress.svc.cluster.local (port 80)

To access your WordPress site from outside the cluster follow the steps below:

1. Get the WordPress URL by running these commands:

  NOTE: It may take a few minutes for the LoadBalancer IP to be available.
        Watch the status with: 'kubectl get svc --namespace wordpress -w wordpress'

   export SERVICE_IP=$(kubectl get svc --namespace wordpress wordpress --template "{{ range (index .status.loadBalancer.ingress 0)
}}{{ . }}{{ end }}")
   echo "WordPress URL: http://$SERVICE_IP/"
   echo "WordPress Admin URL: http://$SERVICE_IP/admin"

2. Open a browser and access WordPress using the obtained URL.

3. Login with the following credentials below to see your blog:

  echo Username: wpcliadmin
  echo Password: $(kubectl get secret --namespace wordpress wordpress -o jsonpath="{.data.wordpress-password}" | base64 -d)

WARNING: There are "resources" sections in the chart not set. Using "resourcesPreset" is not recommended for production. For produc
tion installations, please set the following values according to your workload needs:
  - resources
+info https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/
```

Figure 11: WordPress installation via Helm

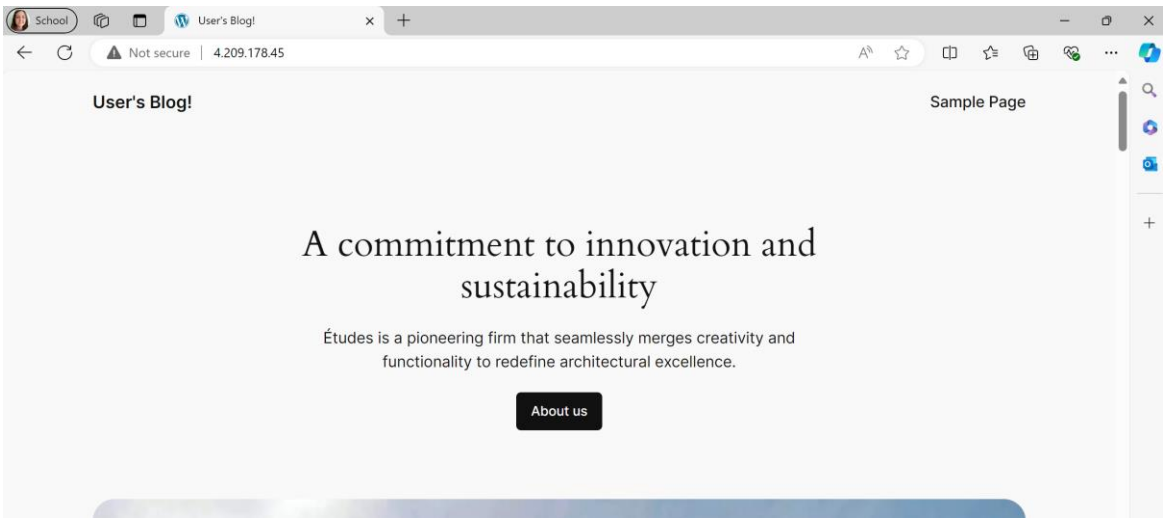

Figure 12: Exposed WordPress application

3.6 Monitoring tools

On the Azure portal, the AKS cluster is visible from the resources. Once clicked on this item for more details, the first tab displays monitoring insights. From this option, Prometheus can be configured as well as Grafana. Figure 13 displays these services. It is important to note that however Grafana can be expensive as it can manage a high volume of real-time data.

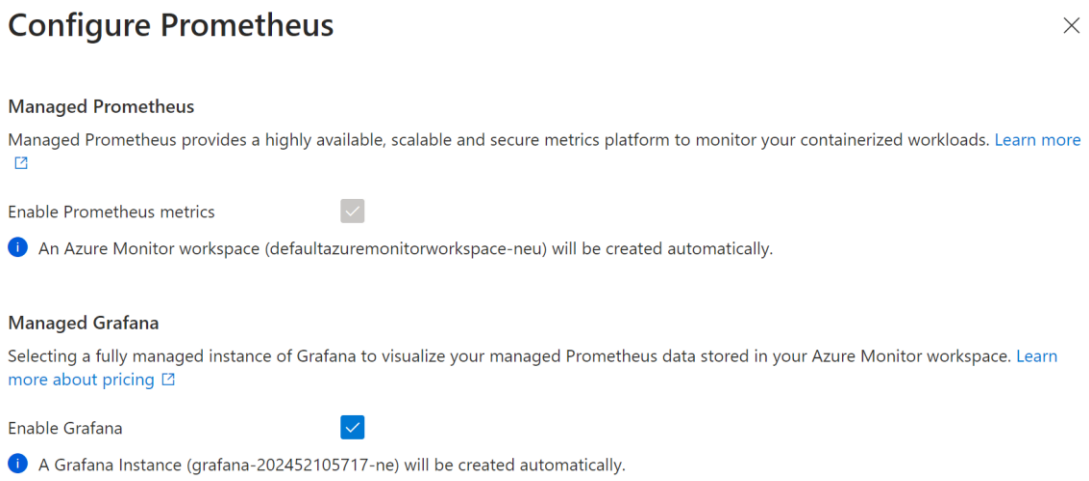

Figure 13: Prometheus and Grafana configuration

Grafana is accessible from Azure resources. Once clicked on it, it is possible to explore its dashboard. Figure 14 presents Grafana from the browser. Kubernetes resources are available from the following path on Grafana dashboard: *Home* > *Dashboard* > *Azure Managed Prometheus* > *Kubernetes / Compute Resources / Pods*.

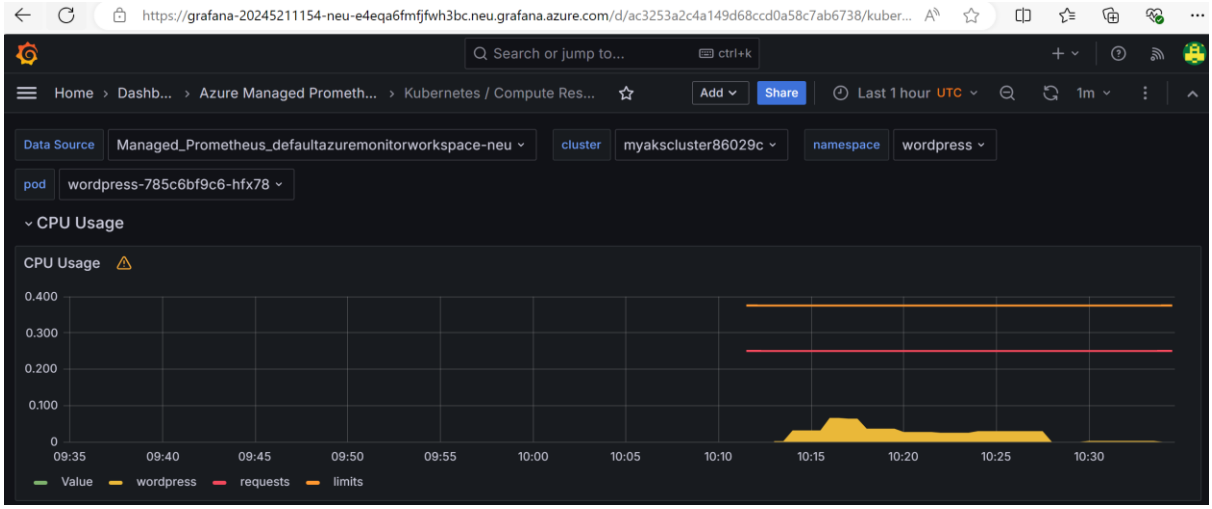

Figure 14: Prometheus and Grafana configuration

On the cluster's page, diagnostic settings such as Kubernetes API Server, Kubernetes Audit and Kubernetes Audit Admin Logs can also be added. They provide significant information from the AKS control plane components (Singh, 2022). To enable this data, the followed tabs must be followed: *Cluster* > *Monitoring* > *Diagnostic settings*. Then, as presented in Figure 15, on the creation of a diagnostic setting, the logs categories Kubernetes API Server, Kubernetes Audit and Kubernetes Audit Admin Logs must be selected.

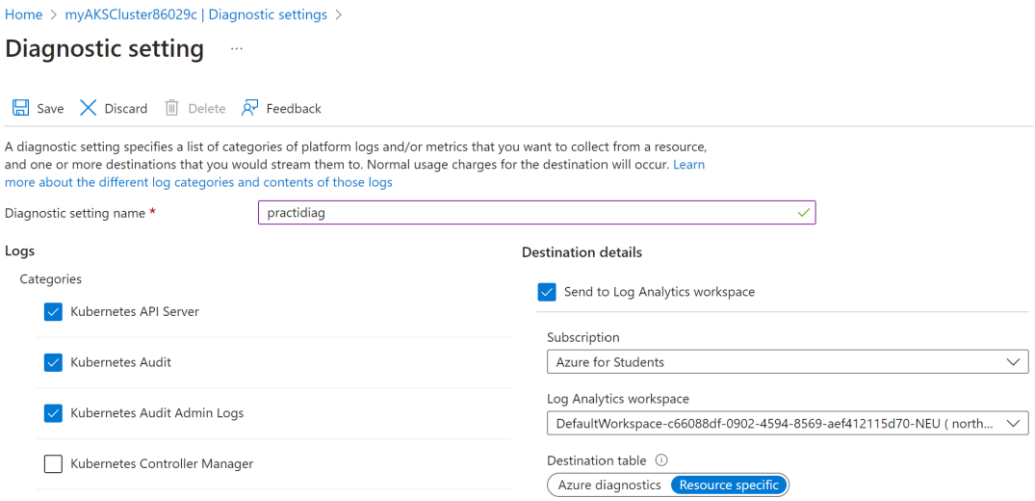

Figure 15: Kubernetes audit configuration

4 Attack simulations

Once the infrastructure and the WordPress application are installed and configured, attacks can then be launched to these vulnerable targets. These are brute-force and DoS attacks.

4.1 Brute-force attack

Before performing a brute-force attack on the exposed WordPress application, a new user with a weak password must be created as shown in Figure 16.

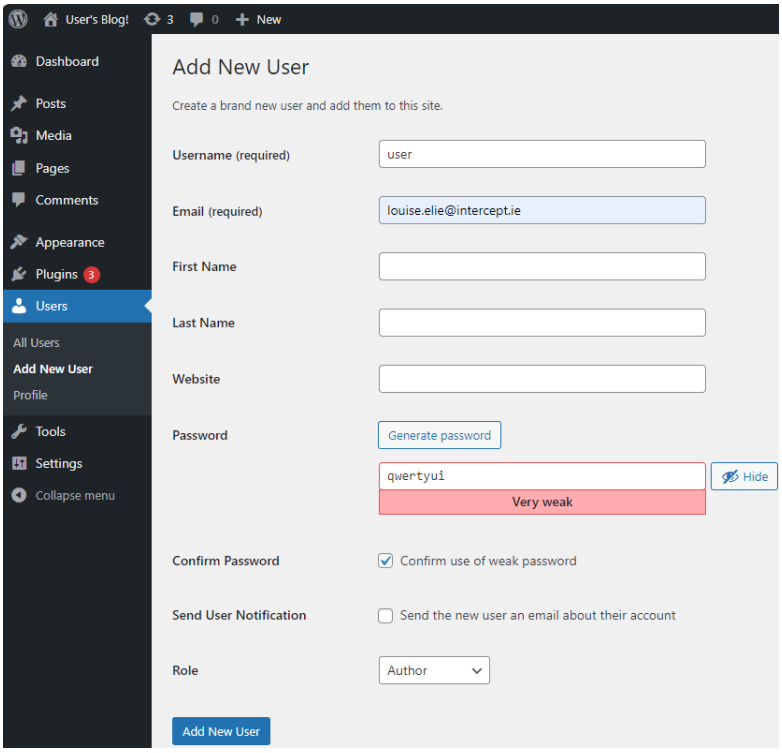

Figure 16: WordPress user creation

However, this user must publish an article on the WordPress site to be listed among the users. Indeed, on the attacker machine, two users are recognised with the following URL: *<wordpress_ip_addess>/wp-json/wp/v2/users*. Figure 17 displays the curl command.

Figure 17: WordPress users list from the attacker machine

The target users have been identified, and to perform the attack requires a list of passwords to test on them. Numerous websites offer these features, the 500 worst passwords[9] should contain our user's password to ensure the success of the attack.

On the attacker's machine, X Brute Forcer is chosen to perform the brute-force attack. The tool is cloned from its GitHub page with the command `git clone https://github.com/bibortone/XBruteForcer.git`. The X Brute Forcer's folder should contain the website's URL in the `list.txt` file and the 500 passwords in the `password.txt` file. To launch the tool, Figure 17 presents the output of the `perl XBruteForcer.pl -l list.txt -p passwords.txt` command. Then, WordPress must be selected to start the brute-force attack in Figure 18. Figure 19 displays the successful attempt of the brute-force attack.

Figure 18: WordPress selection on X Brute Forcer

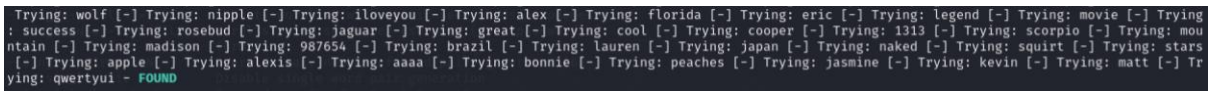

Figure 19: Successful attempt of the brute force attack

[9] https://www.skullsecurity.org/wiki/Passwords

4.2 DoS attack

Apache JMeter is the tool chosen to simulate a DoS attack on the exposed WordPress site. It is installed on the attacker's machine using the download page[10]. The files inside the ZIP folder then need to be extracted. Once the tool is downloaded, Apache JMeter GUI is launched by typing `./jmeter` on the */apache-jmeter-5.6.3/bin* folder. Figure 20 displays the output of the command, Figure 21 the Apache JMeter GUI mode.

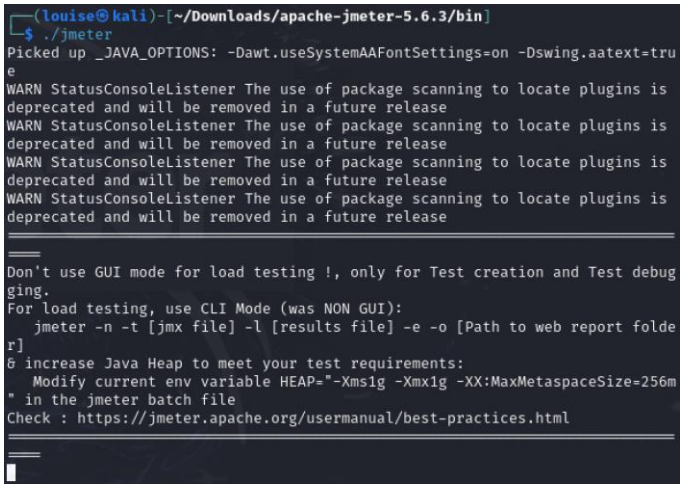

Figure 20: Launch of Apache JMeter

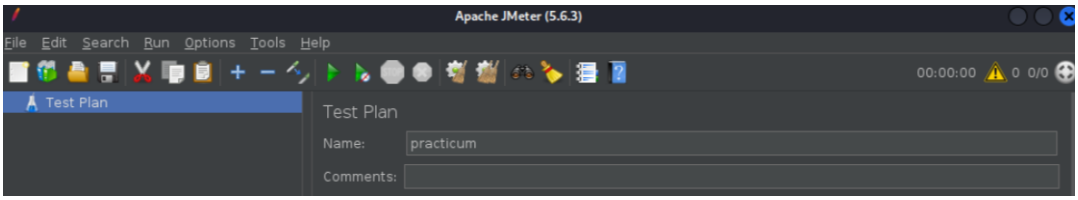

Figure 21: Creation of a new test plan

To perform the DoS attack, a thread group is added to the new test plan. On Figure 22, a simulation of 3000 users were set.

[10] https://github.com/apache/jmeter

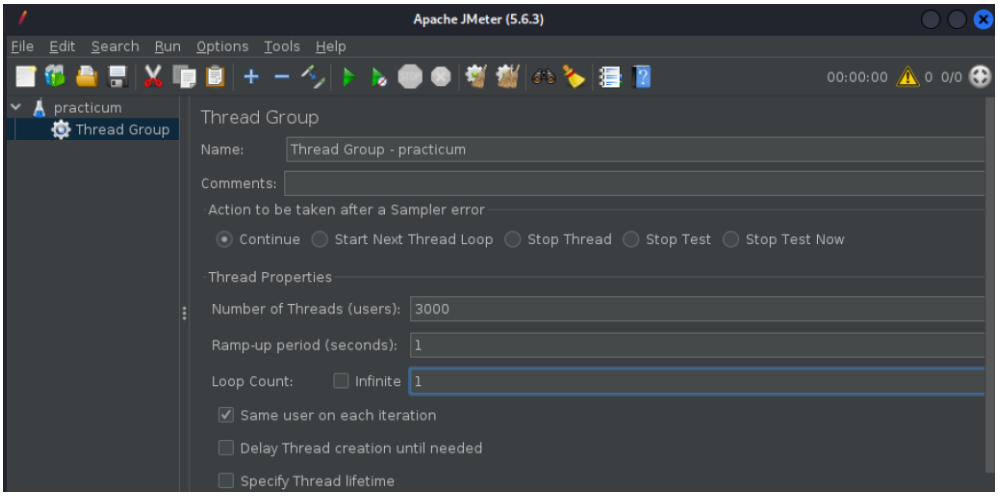

Figure 22: Thread group configuration

This thread group requires a sampler, which are HTTP requests. The server's name and the file path can be specified by right-clicking on *Thread Group*, then selecting *Add > Sampler > HTTP Request*. As seen in Figure 23, the server's name is the WordPress site IP address, and the file path */wp-login.php*.

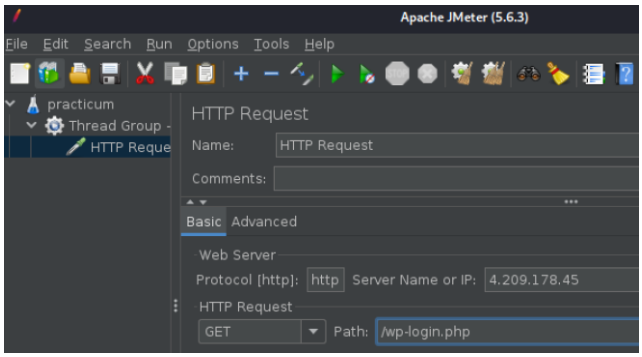

Figure 23: Launch of Apache JMeter

To perform the DoS attack, a result tree listener must be added to the test plan by right-clicking on *HTTP Request*, and then selecting *Listener > View Results Tree*. The DoS attack is initiated by clicking the start button at the top of Figure 24. On the right side of the interface, there is real-time information on the time chronometer and the thread number. At the bottom, a list of the HTTP requests is displayed.

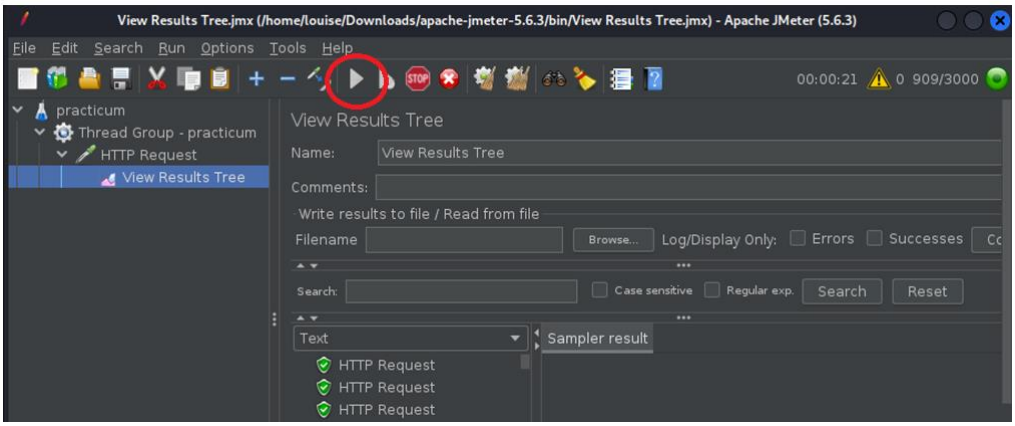

Figure 24: DoS Results Tree

During the attack, the web application becomes available. Figure 25 illustrated the status of the web application at that time.

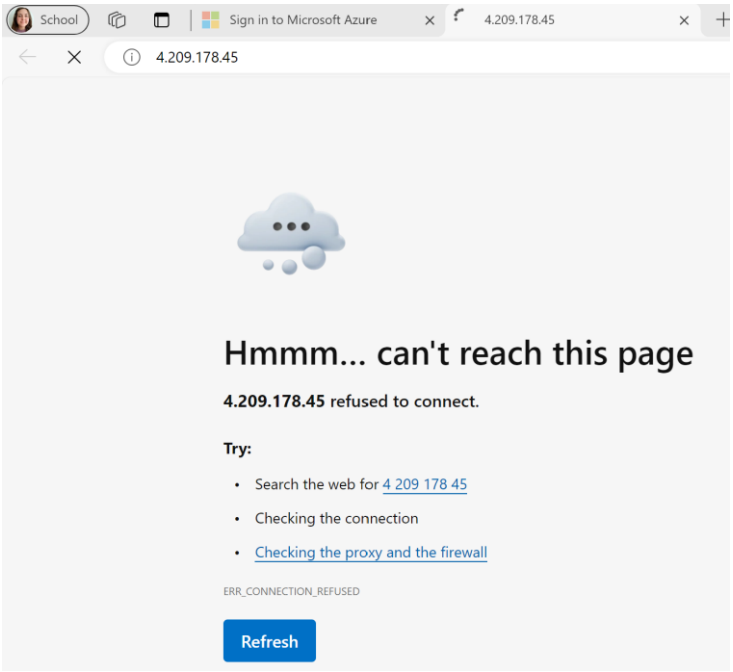

Figure 25: Unreachable web application

5 Data collection

In line with the research hypothesis, the objective is to confirm that containers runtime events and metadata can contribute to the optimisation of digital forensics. After launching the brute force and DoS attacks, data must be collected from these components to gather evidence.

5.1 Container runtime events

Container runtime events can be collected from containers logs, the container orchestration tool (Kubernetes) events and audit logs, and Prometheus metrics on Grafana.

On the Azure Cloud Shell, containers logs are collected with the command `kubectl logs wordpress-785c6bf9c6-zt58v -n wordpress`. Figure 26 displays the output of the command following the brute force attack. These logs are also accessible on the container live logs from the Azure Portal, as presented in Figure 27. In comparison, with live logs, the number of events is displayed.

```
10.177.0.39 - - [23/Jul/2024:13:00p-login.php HTTP/1.1" 200 4133
10.177.0.39 - - [23/Jul/2024:13:02:28 +0000] "GET /wp-login.php HTTP/1.1" 200 4133
10.177.0.39 - - [23/Jul/2024:13:02:38 +0000] "GET /wp-login.php HTTP/1.1" 200 4133
10.177.0.39 - - [23/Jul/2024:13:02:48 +0000] "GET /wp-login.php HTTP/1.1" 200 4133
10.177.0.39 - - [23/Jul/2024:13:02:58 +0000] "GET /wp-login.php HTTP/1.1" 200 4133
10.177.0.39 - - [23/Jul/2024:13:03:08 +0000] "GET /wp-login.php HTTP/1.1" 200 4133
10.177.0.39 - - [23/Jul/2024:13:03:18 +0000] "GET /wp-login.php HTTP/1.1" 200 4133
10.177.0.39 - - [23/Jul/2024:13:03:28 +0000] "GET /wp-login.php HTTP/1.1" 200 4133
10.177.0.39 - - [23/Jul/2024:13:03:38 +0000] "GET /wp-login.php HTTP/1.1" 200 4133
10.177.0.39 - - [23/Jul/2024:13:03:48 +0000] "GET /wp-login.php HTTP/1.1" 200 4133
10.177.0.39 - - [23/Jul/2024:13:03:58 +0000] "GET /wp-login.php HTTP/1.1" 200 4133
10.177.0.39 - - [23/Jul/2024:13:04:08 +0000] "GET /wp-login.php HTTP/1.1" 200 4133
10.177.0.39 - - [23/Jul/2024:13:04:08 +0000] "GET /wp-includes/blocks/navigation/view.min.js?ver=6.5.4 HTTP/1.1" 200 1135
10.177.0.39 - - [23/Jul/2024:13:04:08 +0000] "GET /wp-includes/js/dist/interactivity.min.js?ver=6.5.4 HTTP/1.1" 200 13147
10.177.0.39 - - [23/Jul/2024:13:04:08 +0000] "GET /wp-includes/blocks/navigation/style.min.css?ver=6.5.4 HTTP/1.1" 200 2290
10.177.0.10 - - [23/Jul/2024:13:04:08 +0000] "GET /wp-includes/blocks/image/style.min.css?ver=6.5.4 HTTP/1.1" 200 1597
10.177.0.10 - - [23/Jul/2024:13:04:09 +0000] "GET /wp-includes/js/wp-emoji-release.min.js?ver=6.5.4 HTTP/1.1" 200 5062
10.177.0.39 - - [23/Jul/2024:13:04:18 +0000] "GET /wp-login.php HTTP/1.1" 200 4133
10.177.0.39 - - [23/Jul/2024:13:04:28 +0000] "GET /wp-login.php HTTP/1.1" 200 4133
```

Figure 26: Containers logs with kubectl command

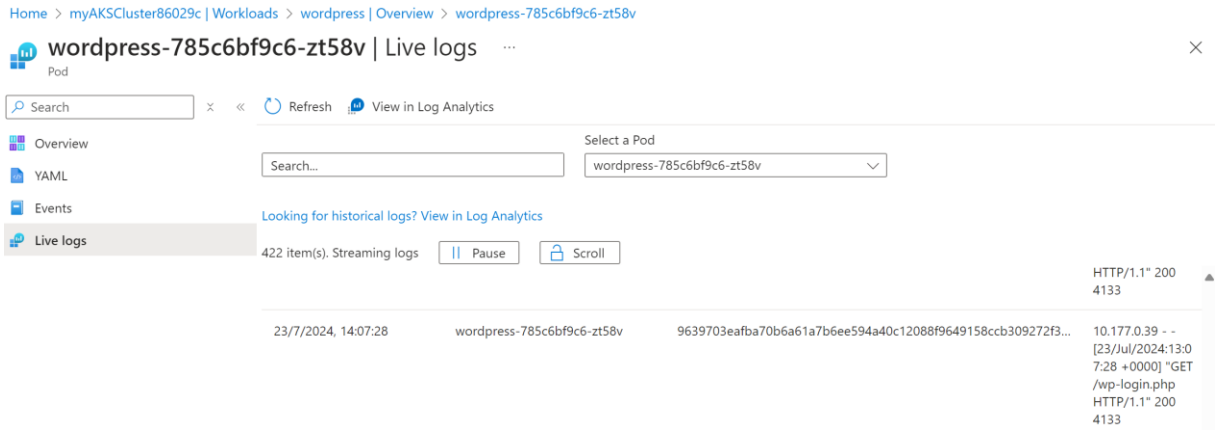

Figure 27: Container live logs from Azure portal

In addition, events from the Kubernetes cluster can be captured with the command `kubectl get events`. Figure 28 displays the output of the command following the DoS attack.

```
louise [ ~ ]$ kubectl get events
LAST SEEN   TYPE      REASON      OBJECT                                MESSAGE
10s         Normal    Pulled      pod/wordpress-785c6bf9c6-vw6gl   Container image "docker.io/bitnami/wordpress:6.5.4-debian-12-r5"
 already present on machine
10s         Normal    Created     pod/wordpress-785c6bf9c6-vw6gl   Created container wordpress
10s         Normal    Started     pod/wordpress-785c6bf9c6-vw6gl   Started container wordpress
28s         Warning   Unhealthy   pod/wordpress-785c6bf9c6-vw6gl   Readiness probe failed: Get "http://10.177.0.45:8080/wp-login.ph
p": context deadline exceeded (Client.Timeout exceeded while awaiting headers)
98s         Warning   Unhealthy   pod/wordpress-785c6bf9c6-vw6gl   Liveness probe failed: dial tcp 10.177.0.45:8080: i/o timeout
92s         Warning   Unhealthy   pod/wordpress-785c6bf9c6-vw6gl   Readiness probe failed: Get "http://10.177.0.45:8080/wp-login.ph
p": dial tcp 10.177.0.45:8080: connect: connection refused
92s         Warning   Unhealthy   pod/wordpress-785c6bf9c6-vw6gl   Liveness probe failed: dial tcp 10.177.0.45:8080: connect: conne
ction refused
22s         Warning   BackOff     pod/wordpress-785c6bf9c6-vw6gl   Back-off restarting failed container wordpress in pod wordpress-
785c6bf9c6-vw6gl_wordpress(0a3bacb8-dd74-4484-a891-9623d587c7d4)
```

Figure 28: Recent events on the Kubernetes cluster

Grafana can be used to analyse resources metrics to trace back events. Indeed, Figure 29 indicates resource spikes during attacks.

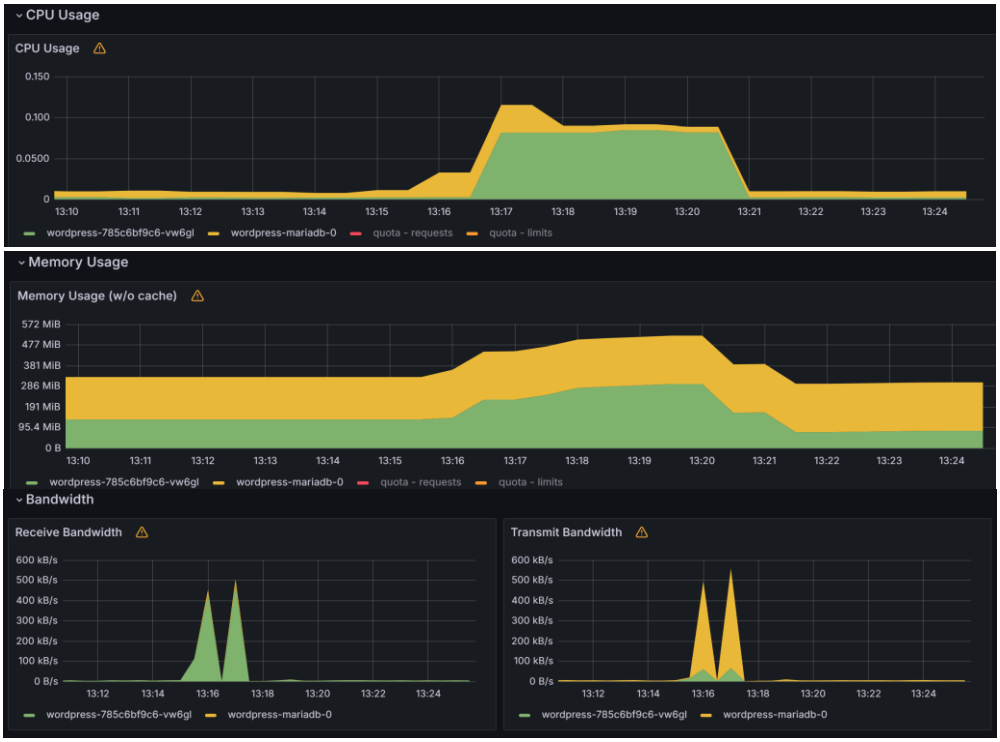

Figure 28: Resource spikes on Grafana

Similarly to containers logs, audit Kubernetes logs also contain information. Audit logs are managed by Azure and cannot be collected directly from the cluster. As presented in Figure 29, audit logs are visible using queries from the “Logs” tab on the AKS cluster’s page.

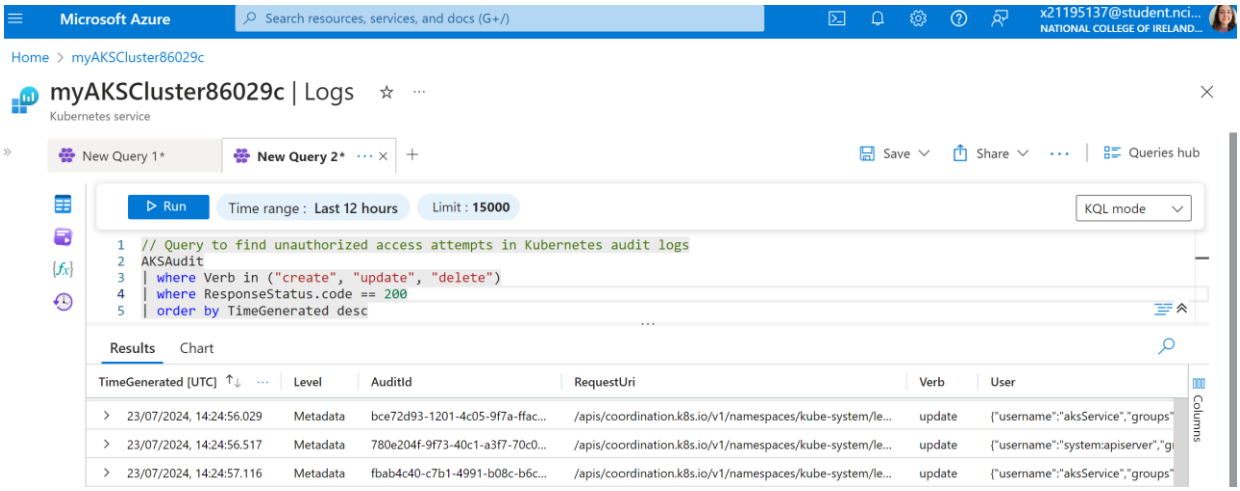

Figure 29: Kubernetes audit logs

5.2 Container metadata

Container metadata describe information of a resource. In this project, it can be related to object and node metadata.

Kubernetes object metadata such as pods and services can be collected with the following command: `kubectl get <object>`. For example, in Figure 30, pod restarts are a sign of a potential DoS attack.

Figure 30: Pod restarts

Once the kubectl get command identifies the object, the kubectl describe command provides further detailed information including node associated, IP address or even events. Figure 31 shows the output of `kubectl describe pods <pod_name> -n <namespace> command.`

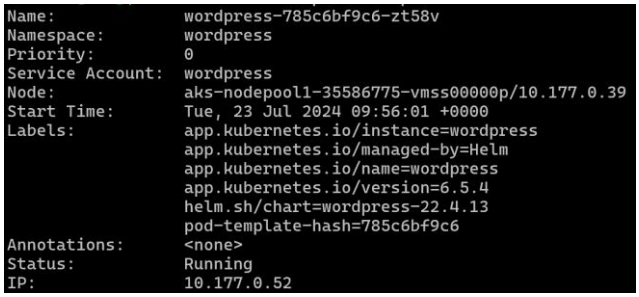

Figure 31: Pod metadata

Figure 32 also displays information on the WordPress service with the output of `kubectl describe services wordpress -n wordpress command`.

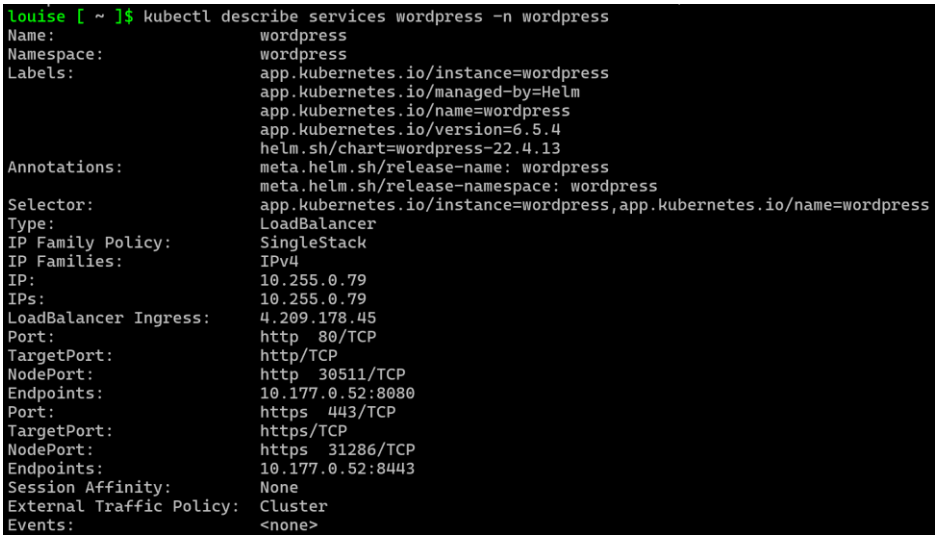

Figure 32: Service metadata

It is also possible to capture significant information from the nodes such as the resource utilisation and events. After identifying the nodes with the `kubectl get nodes` command, the `kubectl describe nodes <node_name>` command provides insights into the allocated resources for each container and overall resource usage, as shown in Figure 33, along with events. For instance, Figure 34 displays killed processes, which could be a potential sign of a DoS attack.

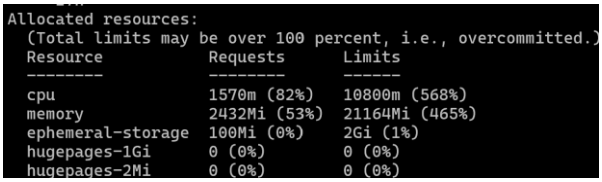

Figure 33: Node metadata

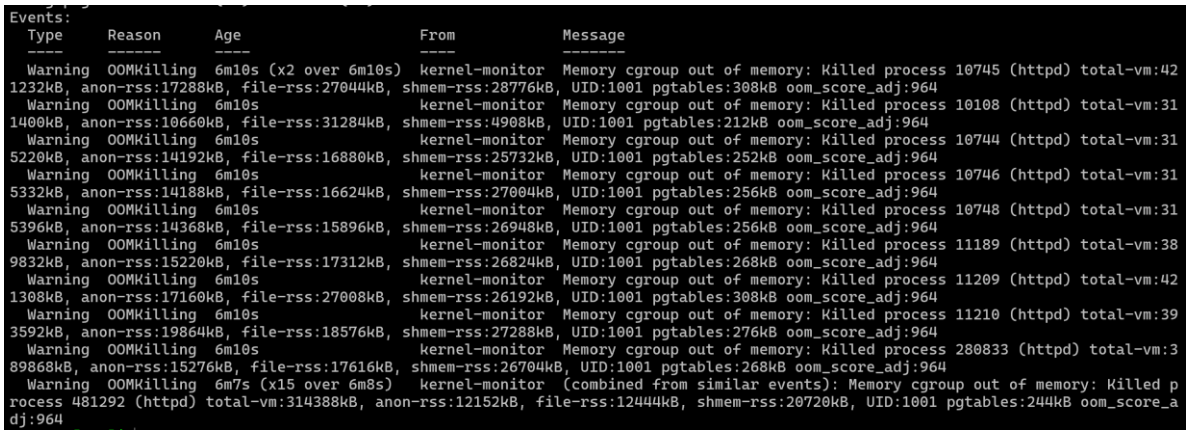

Figure 34: Node events

6 Hardening solutions

To defend against these two types of attack, four mitigation solutions were implemented: a limit login WordPress add-on, rate limiting, IP blocking, and alerts on login activities and resource utilisation.

6.1 Limit login

The WPS Limit Login add-on[11] on WordPress is widely used to limit login attempts on a WordPress site. On the admin page at *<wordpress_ip_or_url>/wp-admin*, the add-on can be downloaded from the Plugins tab on Figure 35.

[11] https://wordpress.org/plugins/wps-limit-login/

Figure 35: Limit login attempts add-on installation

During the installation, as presented in Figure 36, the default configuration is chosen to limit to 3 retries for a period of 20 minutes.

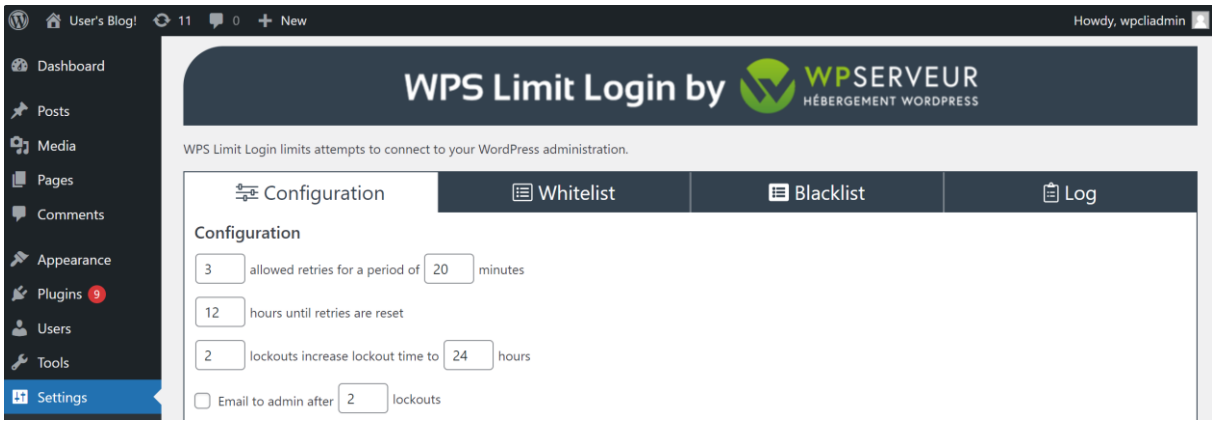

Figure 36: Limit of login attempts parameters

After another brute-force attack, Figure 37 below confirms the unsuccessful login attempt, and thus Figure 38 the locked login page.

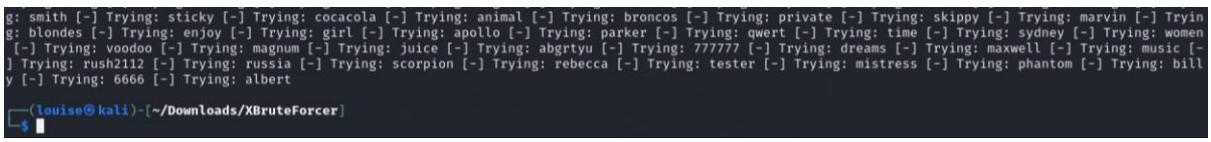

Figure 37: Login attempts with X Brute Forcer

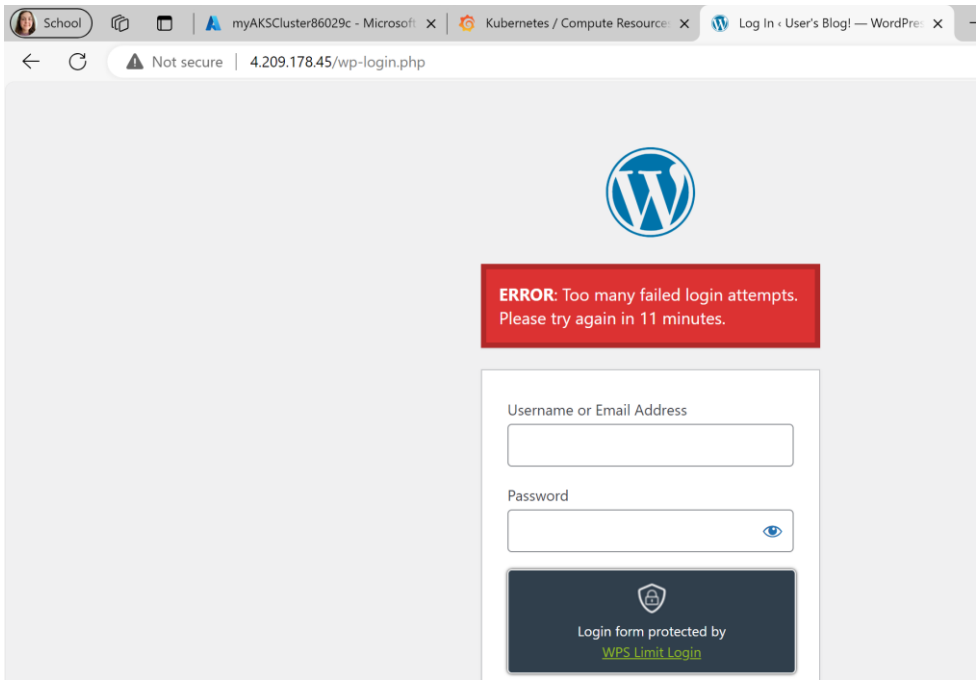

Figure 38: Locked login page

6.2 Rate limiting and IP blocking

To mitigate DoS attacks on the Kubernetes environment, two options were implemented on the ingress controller: rate limiting and IP blocking.

On Figure 39, the existing NGINX ingress was edited to add parameter regarding rate limiting on the metadata section. Connections are limited to 10 IP addresses and 5 requests per second. Once the file is saved, the modifications are applied with the following command: `kubectl apply -f ingress.yaml`.

```
 GNU nano 6.0                                          ingress.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: wordpress-ingress
  annotations:
    nginx.ingress.kubernetes.io/limit-connections: "10"
    nginx.ingress.kubernetes.io/limit-rps: "5"
    nginx.ingress.kubernetes.io/limit-burst-multiplier: "3"
spec:
  ingressClassName: nginx
  rules:
  - host: 4.209.178.45.nip.io  # Use nip.io as we don't have a domain
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: wordpress
            port:
              number: 80
```

Figure 39: Rate limiting configuration

On the attacks performed, repetitive IP addresses have been observed. A network policy can block those IP addresses. Figure 40 displays the rule. The modification of the file will be applied with the `kubectl apply -f netpolicy.yaml` command.

```
louise [ ~/practikube ]$ cat netpolicy.yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: block-suspicious-ip
  namespace: wordpress
spec:
  podSelector:
    matchLabels:
      app: wordpress
  policyTypes:
  - Ingress
  ingress:
  - from:
    - ipBlock:
        cidr: "10.177.0.39/32"
    - ipBlock:
        cidr: "10.177.0.10/32"
```

Figure 40: IP blocking rule

Figure 40 presents the NGINX ingress controller logs during a DoS attack, and it shows that the WordPress server was down numerous times. The command used is `kubectl logs -l app.kubernetes.io/name=ingress-nginx -n ingress-nginx`.

```
louise [ ~ ]$ kubectl logs -l app.kubernetes.io/name=ingress-nginx -n ingress-nginx
W0729 12:13:19.422940       7 controller.go:1213] Service "wordpress/wordpress" does not have any active Endpoint.
W0729 12:13:32.916953       7 controller.go:1213] Service "wordpress/wordpress" does not have any active Endpoint.
W0729 12:13:36.251157       7 controller.go:1213] Service "wordpress/wordpress" does not have any active Endpoint.
W0729 12:13:39.584792       7 controller.go:1213] Service "wordpress/wordpress" does not have any active Endpoint.
10.177.0.10 - - [29/Jul/2024:12:54:55 +0000] "POST /cgi-bin/.%2e/.%2e/.%2e/.%2e/.%2e/.%2e/.%2e/.%2e/.%2e/bin/sh HTTP/1.1" 400
150 "-" "-" 81 0.114 [] [] - - - - 66b5a66af181418af3cf44a92390376f
10.177.0.10 - - [29/Jul/2024:12:54:56 +0000] "POST /cgi-bin/%%32%65%%32%65/%%32%65%%32%65/%%32%65%%32%65/%%32%65%%32%65/%%32%65%%32
%65/%%32%65%%32%65/%%32%65%%32%65/bin/sh HTTP/1.1" 400 150 "-" "-" 136 0.118 [] [] - - - - 1adc1b6cb27c1971f9a161d7807e81a8
W0729 13:27:35.857692       7 controller.go:1213] Service "wordpress/wordpress" does not have any active Endpoint.
W0729 13:28:37.947693       7 controller.go:1213] Service "wordpress/wordpress" does not have any active Endpoint.
10.177.0.10 - - [29/Jul/2024:13:35:47 +0000] "POST /cgi-bin/.%2e/.%2e/.%2e/.%2e/.%2e/.%2e/.%2e/.%2e/.%2e/bin/sh HTTP/1.1" 400
150 "-" "-" 81 0.413 [] [] - - - - 8324f90889020137 48d1f6dc7dc3fc5d
10.177.0.39 - - [29/Jul/2024:13:35:49 +0000] "POST /cgi-bin/%%32%65%%32%65/%%32%65%%32%65/%%32%65%%32%65/%%32%65%%32%65/%%32%65%%32
%65/%%32%65%%32%65/%%32%65%%32%65/bin/sh HTTP/1.1" 400 150 "-" "-" 136 0.417 [] [] - - - - 1bb80e78657ef8567559347ba9698a95
```

Figure 41: Ingress logs

6.3 Alerts

To improve detection time, several alerts were created following the first attack attempts. This includes login attempts and resource spikes.

An alert can be created through the “Logs” tab on the AKS cluster page. When running queries on the system, it is possible to associate them with alerts. Figure 42 details the rule created for multiple login attempts. If more than 10 login attempts are detected in less than 5 minutes, an email is sent to alert.

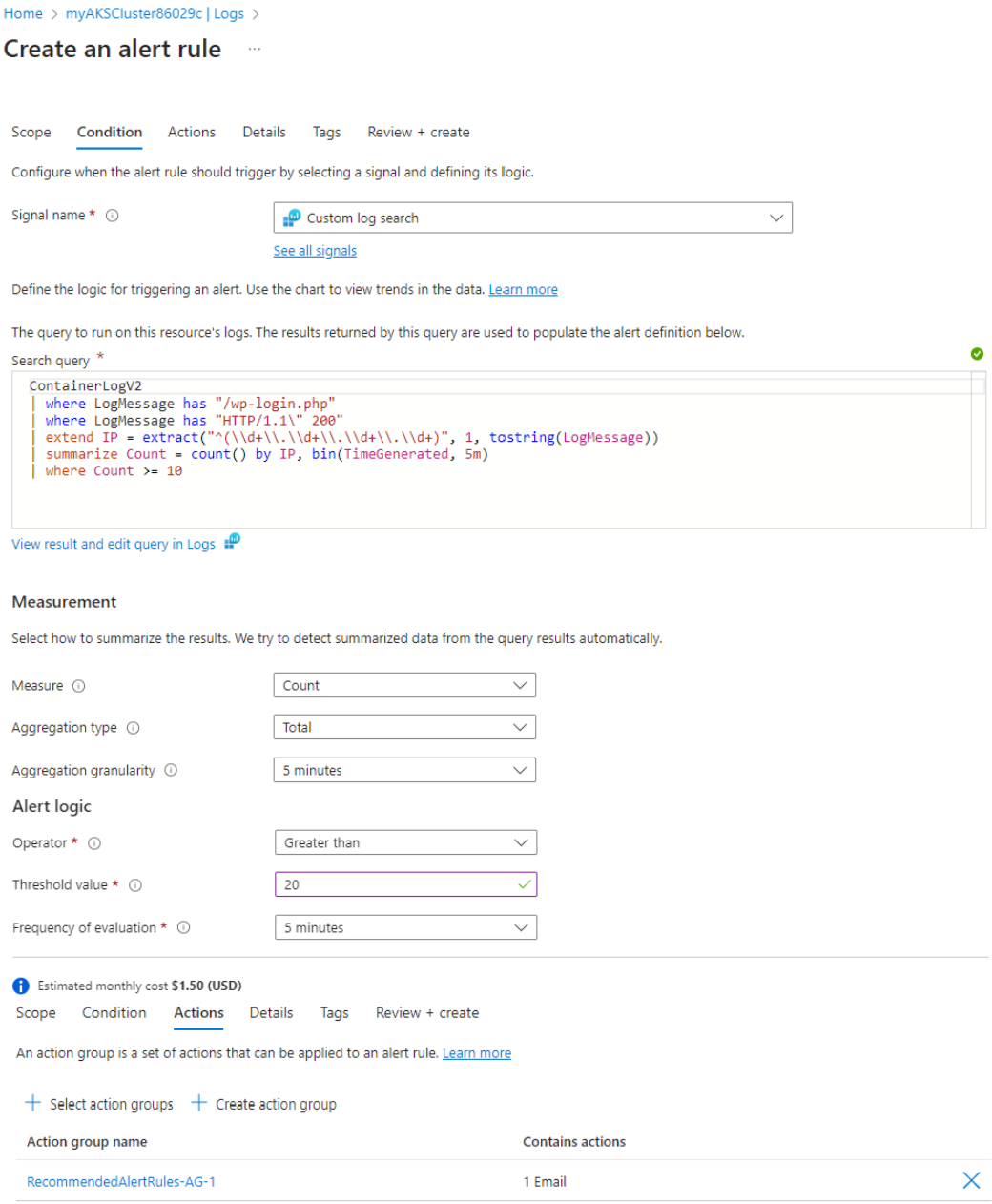

Figure 42: Login attempts alert rule

Regarding resources spikes such as CPU and memory usage, Prometheus provides predefined rules on different Kubernetes levels, from pods to cluster. As presented in Figure 43, the recommended pod level alerts were used for this project.

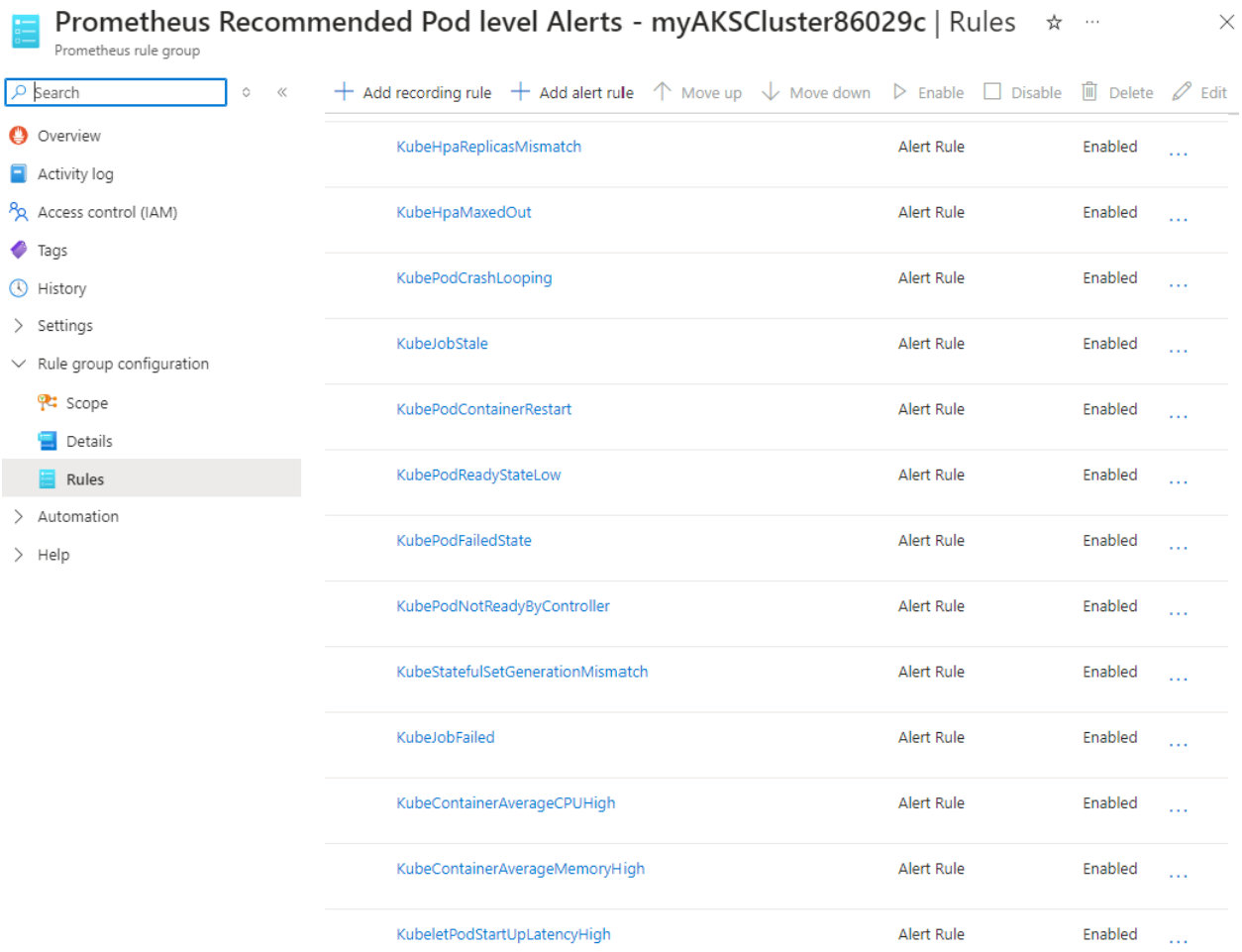

Figure 43: Prometheus preset rules

It contains rules such as alerting CPU usage above 75% as detailed in Figure 44.

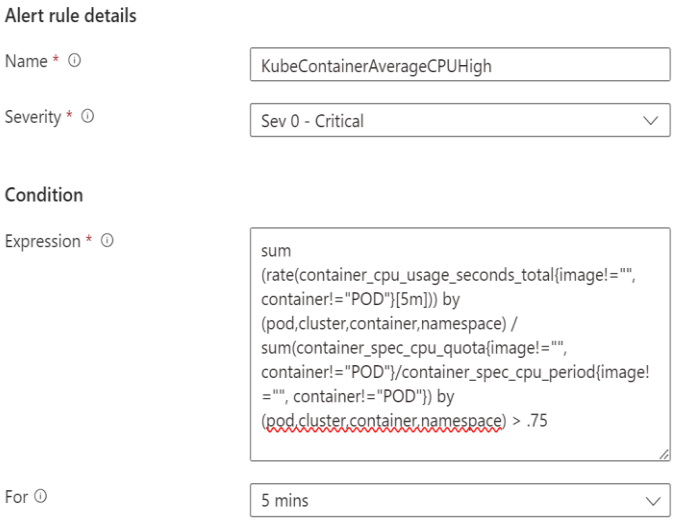

Figure 44: KubeContainerAverageCPUHigh alert rule

Similarly, another rule detects the memory usage and alerts the user if it is above 75% of its overall capacity in Figure 45. The email alert sent to the user is displayed in Figure 46.

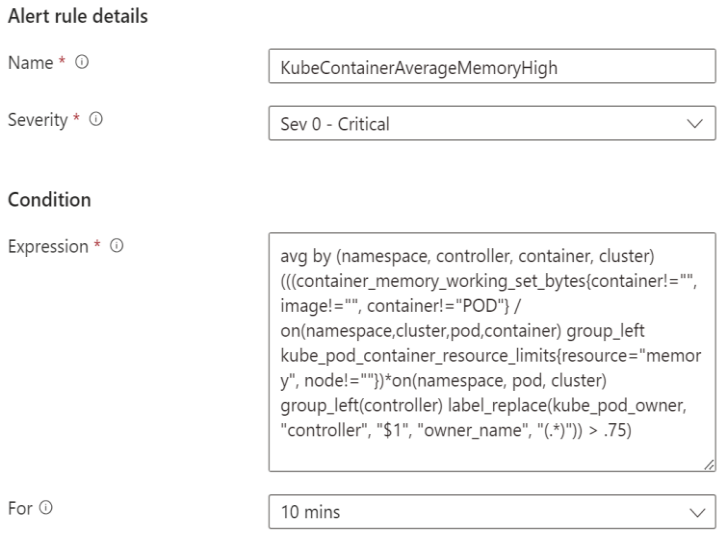

Figure 45: KubeContainerAverageMemoryHigh alert rule

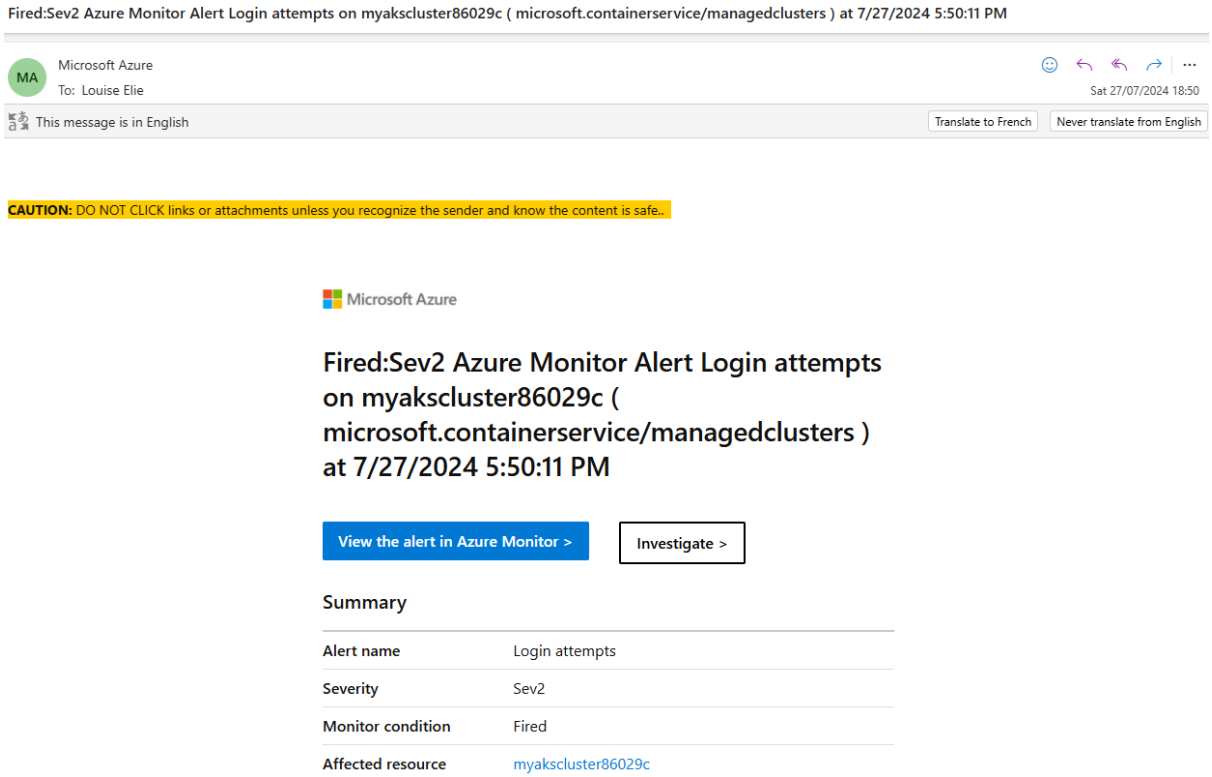

Figure 46: Alert email

References

Microsoft Learn (2024) *Deploy WordPress on AKS cluster by using Azure CLI*. Available at: https://learn.microsoft.com/en-us/azure/mysql/flexible-server/tutorial-deploy-wordpress-on-aks [Accessed 20 June 2024].

Singh, H. (2022) *Investigate actions on Azure Kubernetes Service using Auditing*, *Medium*, 29 September. Available at: https://itnext.io/whodunit-investigate-actions-on-aks-using-auditing-1db3ccf9ae86 [Accessed 20 July 2024].