2024

Assignment 1

MONITORING A KUBERNETES-HOSTED APPLICATION USING SSO WITH GMAIL, SYSLOG, AZURE LOG ANALYTICS, AND GRAFANA

GROUP MEMBERS: BO YUAN, SHIFENG SONG, YINGDA ZHANG, KENAN YUSUF |

PRESENTED TO: ISLAM GOMAA | 03/22/2024

Code Repo

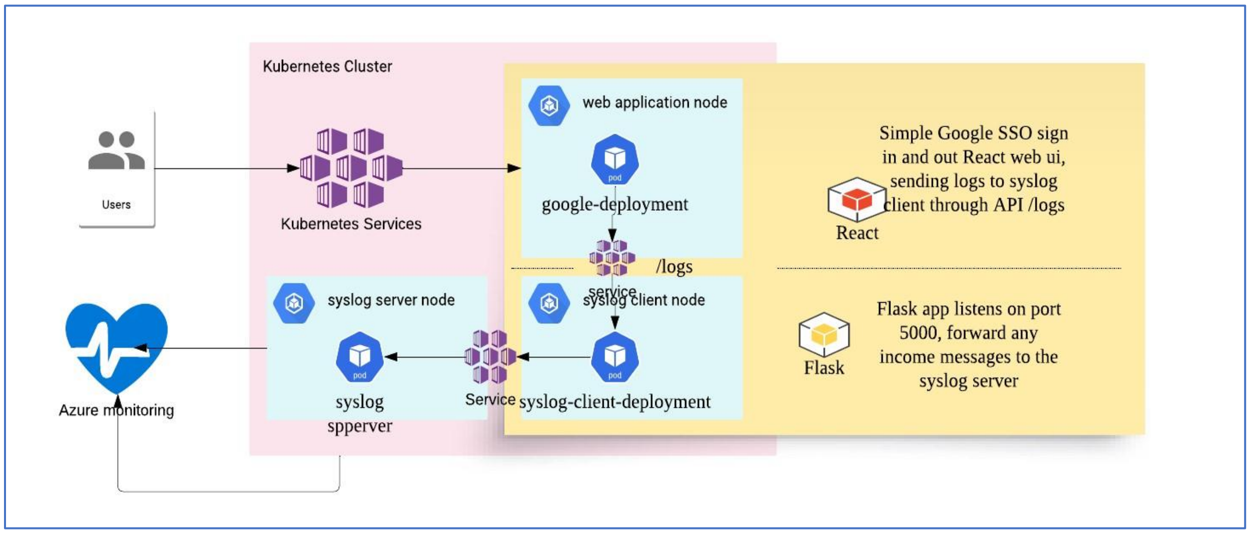
Github Address**:** **https://github.com/luckyeven/k8s-google-sso.git**

Introduction

In this assignment, we tackled a comprehensive exploration of monitoring and managing system logs through Azure Grafana, coupled with the deployment of a React web application that incorporates Google Single Sign-On (SSO) for authentication, all within a Kubernetes environment. Our initial approach was methodical, beginning with the establishment of a syslog server and client, alongside the deployment of the React SSO web page. This foundational step allowed us to delve into the intricacies of Kubernetes services, which facilitated IP routing for each pod, thereby ensuring seamless communication and functionality within our setup. As our investigation progressed, we identified opportunities to refine our project. This led to a focused approach, where we prioritized the deployment of the React web page. A key aspect of our refined strategy involved the direct transmission of syslogs from the cluster to Azure Log Monitor. This decision was not made lightly; it was informed by a desire to streamline our operations and enhance the efficiency of our monitoring processes. By doing so, we were able to gain invaluable insights into the cluster's performance, thereby elevating our understanding of the system's dynamics and operational capabilities.

Our journey through this assignment was characterized by a series of strategic decisions and adjustments, all aimed at optimizing our setup for better performance and more insightful analysis. Through this process, we not only honed our technical skills but also developed a deeper appreciation for the complexities and challenges of managing and monitoring applications in a distributed cloud computing environment. This experience has underscored the importance of flexibility, strategic planning, and continuous learning in this field.

Architecture (Phase I)



This schematic outlines the operational flow of a system designed for log management and application deployment, leveraging the capabilities of Kubernetes and Azure. At the core of this system, users engage with applications hosted on a Kubernetes cluster, where authentication is managed through Google's Single Sign-On (SSO) mechanism.

The Kubernetes cluster serves as the foundational infrastructure, hosting various applications and services within its architecture. It is composed of multiple pods, each capable of mutual communication, thereby fostering an integrated environment for the hosted applications. Networking within this cluster is facilitated by Kubernetes Services, which ensure seamless connectivity among the pods as well as with external networks, thereby extending the cluster's reach beyond its internal confines.
Monitoring and performance analysis of the cluster are conducted through Azure Monitoring. This tool plays a pivotal role in ensuring the optimal functioning of the cluster by providing insights into its performance and health metrics.

Within the confines of the Kubernetes cluster, the deployment unfolds as follows:

- **Web Application Node**: A pod, identified as **google-deployment**, hosts a React web application. This application is distinguished by its implementation of Google SSO, facilitating user authentication. User activities, such as signing in or out, trigger the generation of logs, which are then dispatched to a syslog server through a **/logs** API endpoint.

- **Syslog Client Node**: Operating within the **syslog-client-deployment** service, this pod acts as a syslog client. Its primary function is to capture logs from the web application and relay them to the syslog server, thus serving as an intermediary in the log collection process.
- **Syslog Server Node**: Identified as **syslog-server**, this pod aggregates logs received from the syslog client. It may also engage in further processing of these logs.
- **Flask Component**: A Flask application, listening on port 5000, plays a crucial role in forwarding incoming messages to the syslog server, thus contributing to the log compiling mechanism.

The workflow proceeds as follows: Users access the React web application deployed within the Kubernetes cluster, where Google SSO manages authentication. Subsequent events, such as user sign-ins or sign-outs, prompt the web application to generate and send logs to the syslog client via the **/logs** endpoint. These logs are then collected by the syslog client and forwarded to the syslog server for aggregation and potential processing. The syslog server interfaces with Azure Monitoring, enabling the analysis of the cluster's performance based on the aggregated logs. Through continuous monitoring, Azure Monitoring ensures the cluster's health and performance are maintained at optimal levels, informed by the comprehensive log data received from the syslog server.

Deployments

Deployments of syslog server and client pods

ComputingPost. (n.d.). *Configure rsyslog centralized log server on Ubuntu*

22.04/20.04/18.04. Retrieved from https://computingpost.medium.com/configure-rsysloghttps://computingpost.medium.com/configure-rsyslog-centralized-log-server-on-ubuntu-22-04-20-04-18-04-b5222129b3f3centralized-log-server-on-ubuntu-22-04-20-04-18-04-b5222129b3f3

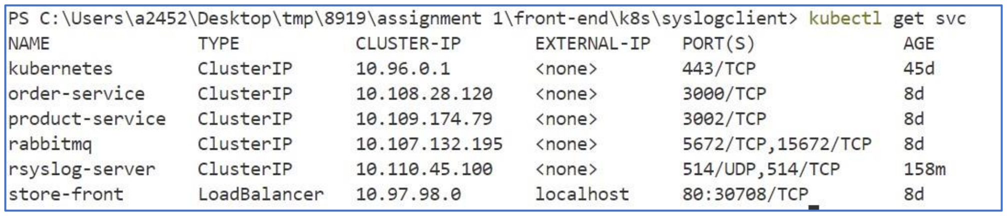
a. Create syslog server pod:

```
kubectl apply -f rsyslog-server-deployment.yaml
```

b. Create syslog server service:

```
kubectl apply -f rsyslog-server-service.yaml
```

c. Get all services on the cluster

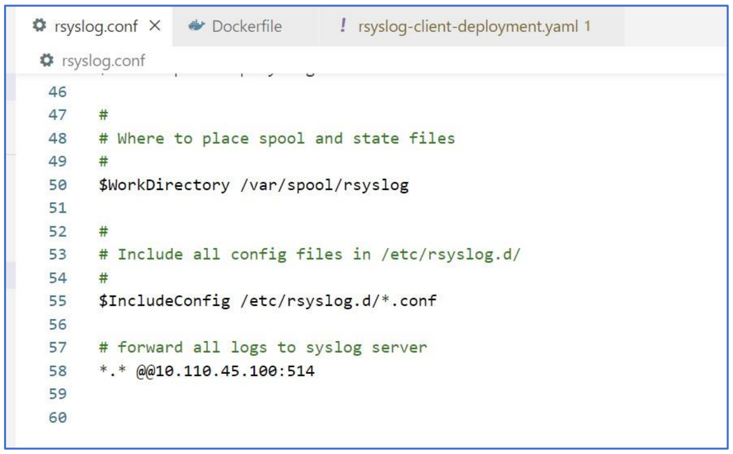


```
PS C:\Users\a2452\Desktop\tmp\8919\assignment 1\front-end\k8s\syslogclient> kubectl get svc
NAME               TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)              AGE
kubernetes         ClusterIP      10.96.0.1        <none>        443/TCP              45d
order-service      ClusterIP      10.108.28.120    <none>        3000/TCP             8d
product-service    ClusterIP      10.109.174.79    <none>        3002/TCP             8d
rabbitmq           ClusterIP      10.107.132.195   <none>        5672/TCP,15672/TCP   8d
rsyslog-server     ClusterIP      10.110.45.100    <none>        514/UDP,514/TCP      158m
store-front        LoadBalancer   10.97.98.0       localhost     80:30708/TCP         8d
```

In this case 10.110.45.100

d. Create syslog client

i. Replace the ip address in the following file (to point to the correct syslog server container)

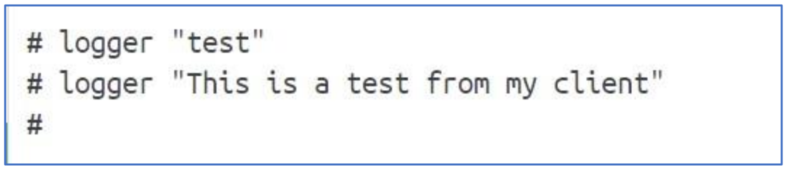


```
#
# Where to place spool and state files
#
$WorkDirectory /var/spool/rsyslog

#
# Include all config files in /etc/rsyslog.d/
#
$IncludeConfig /etc/rsyslog.d/*.conf

# forward all logs to syslog server
*.* @@10.110.45.100:514
```

ii. Apply the deployment of client pod.

```
kubectl apply -f .\rsyslog-client-deployment.yaml
```

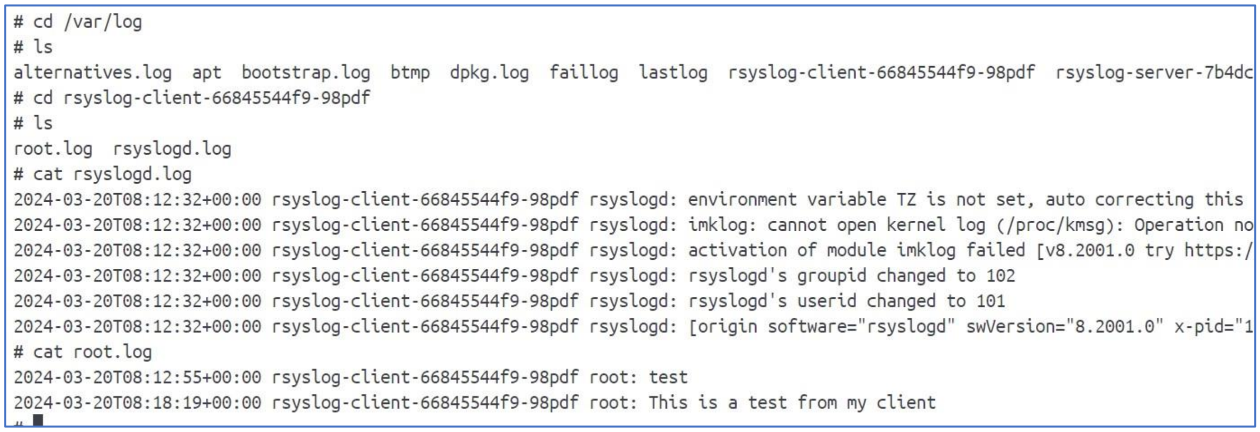
Test: execute this command in syslog client: logger "test”.



```
# logger "test"
# logger "This is a test from my client"
#
```

Then go to the server. CD into the directory cd /var/log , you will see the rsyslog-flient-*** is created.

Navigate into that folder, cat root.log



```
# cd /var/log
# ls
alternatives.log  apt  bootstrap.log  btmp  dpkg.log  faillog  lastlog  rsyslog-client-66845544f9-98pdf  rsyslog-server-7b4dc
# cd rsyslog-client-66845544f9-98pdf
# ls
root.log  rsyslogd.log
# cat rsyslogd.log
2024-03-20T08:12:32+00:00 rsyslog-client-66845544f9-98pdf rsyslogd: environment variable TZ is not set, auto correcting this
2024-03-20T08:12:32+00:00 rsyslog-client-66845544f9-98pdf rsyslogd: imklog: cannot open kernel log (/proc/kmsg): Operation no
2024-03-20T08:12:32+00:00 rsyslog-client-66845544f9-98pdf rsyslogd: activation of module imklog failed [v8.2001.0 try https:/
2024-03-20T08:12:32+00:00 rsyslog-client-66845544f9-98pdf rsyslogd: rsyslogd's groupid changed to 102
2024-03-20T08:12:32+00:00 rsyslog-client-66845544f9-98pdf rsyslogd: rsyslogd's userid changed to 101
2024-03-20T08:12:32+00:00 rsyslog-client-66845544f9-98pdf rsyslogd: [origin software="rsyslogd" swVersion="8.2001.0" x-pid="1
# cat root.log
2024-03-20T08:12:55+00:00 rsyslog-client-66845544f9-98pdf root: test
2024-03-20T08:18:19+00:00 rsyslog-client-66845544f9-98pdf root: This is a test from my client
#
```

Two logs are received.

Deployment of Web-APP Google SSO Pod

1. Google-SSO pod deployment

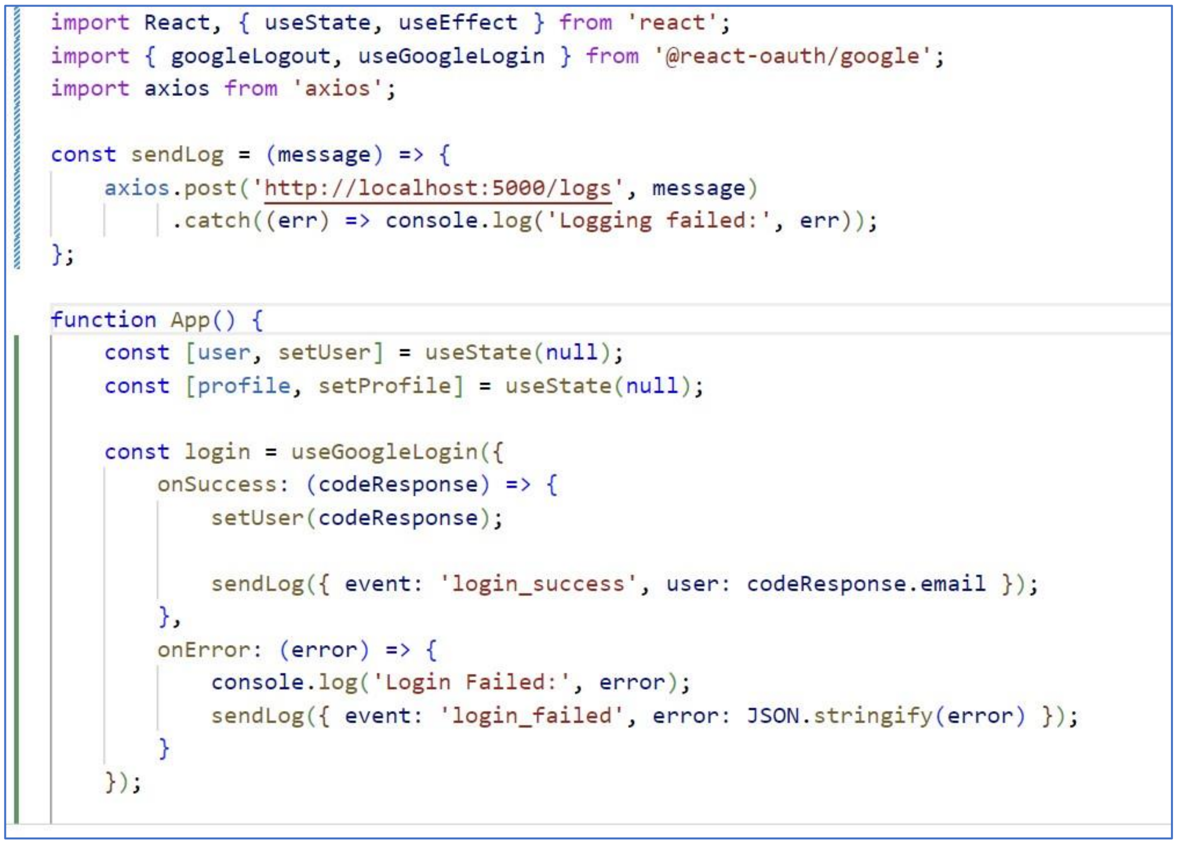
Google sso sign in and out, implemented with React. Also sends logging information to syslog client through API/logs

First step is to get the client id from google by creating a new application.

Muhammed, S. (n.d.). *React JS: A step-by-step guide to Google Authentication*. Retrieved from https://medium.com/@sahadmuhammed289/react-js-a-step-by-step-guide-to-googlehttps://medium.com/@sahadmuhammed289/react-js-a-step-by-step-guide-to-google-authentication-926d0d85edbdauthentication-926d0d85edbd

Steps for deployment.

1. Kubectl get svc
2. Replace address(since i am using docker k8s, which does not have load balancer, so i need to do a port forwarding for the service. In development it is okay, when it comes to production , this should be the actual address of kube services .)



```
import React, { useState, useEffect } from 'react';
import { googleLogout, useGoogleLogin } from '@react-oauth/google';
import axios from 'axios';

const sendLog = (message) => {
    axios.post('http://localhost:5000/logs', message)
        .catch((err) => console.log('Logging failed:', err));
};

function App() {
    const [user, setUser] = useState(null);
    const [profile, setProfile] = useState(null);

    const login = useGoogleLogin({
        onSuccess: (codeResponse) => {
            setUser(codeResponse);

            sendLog({ event: 'login_success', user: codeResponse.email });
        },
        onError: (error) => {
            console.log('Login Failed:', error);
            sendLog({ event: 'login_failed', error: JSON.stringify(error) });
        }
    });
```

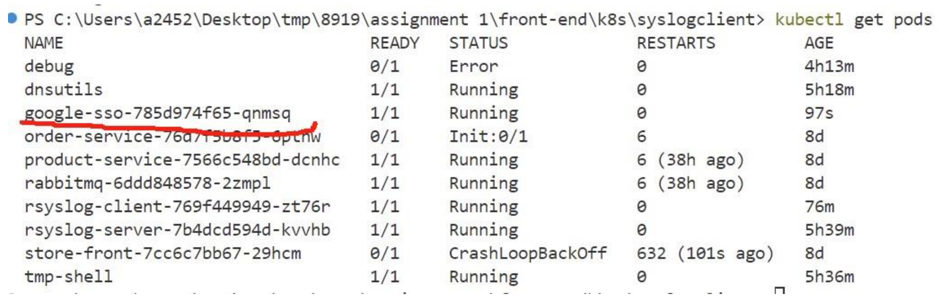
3. Build docker file.
4. Push docker file.

5. kubecrl apply google-sso-deployment.yaml (change the image with your own)

6. kubectl apply google-sso-service.yaml

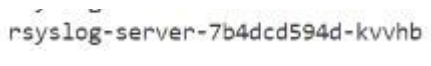
7. Because docker K8s doesn't provide external ip address so we need to forward the port

   a. Find the pod name using kubectl get pods

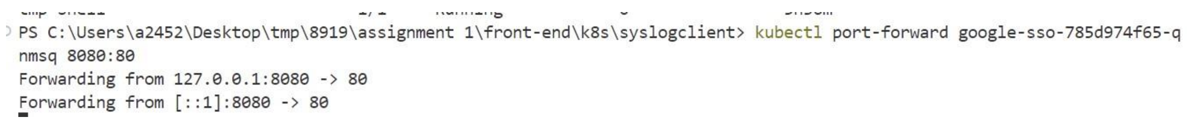


```
PS C:\Users\a2452\Desktop\tmp\8919\assignment 1\front-end\k8s\syslogclient> kubectl get pods
NAME                                READY   STATUS             RESTARTS        AGE
debug                               0/1     Error              0               4h13m
dnsutils                            1/1     Running            0               5h18m
google-sso-785d974f65-qnmsq         1/1     Running            0               97s
order-service-76d7f5b8f5-6ptnw      0/1     Init:0/1           6               8d
product-service-7566c548bd-dcnhc    1/1     Running            6 (38h ago)     8d
rabbitmq-6ddd848578-2zmpl           1/1     Running            6 (38h ago)     8d
rsyslog-client-769f449949-zt76r     1/1     Running            0               76m
rsyslog-server-7b4dcd594d-kvvhb     1/1     Running            0               5h39m
store-front-7cc6c7bb67-29hcm        0/1     CrashLoopBackOff   632 (101s ago)  8d
tmp-shell                           1/1     Running            0               5h36m
```

   b. We need to do port forwarding for syslog-client-service as well. Again this is not a good practice in production,but in development its okay.

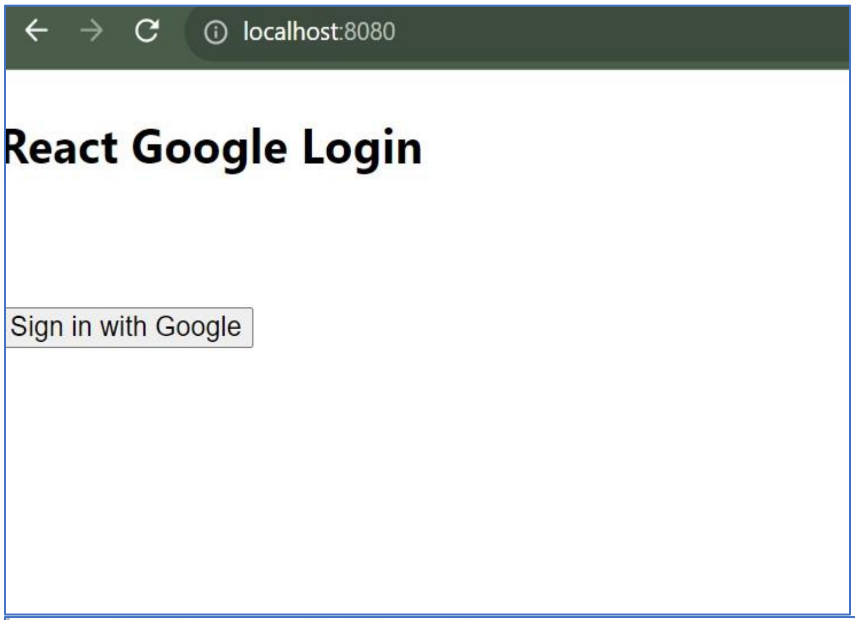
   The service pod name is  `rsyslog-server-7b4dcd594d-kvvhb` .

8. kubectl port-forward google-sso-785d974f65-t6qpf 8080:80 kubectl port-forward rsyslog-server-7b4dcd594d-kvvhb 5000:80



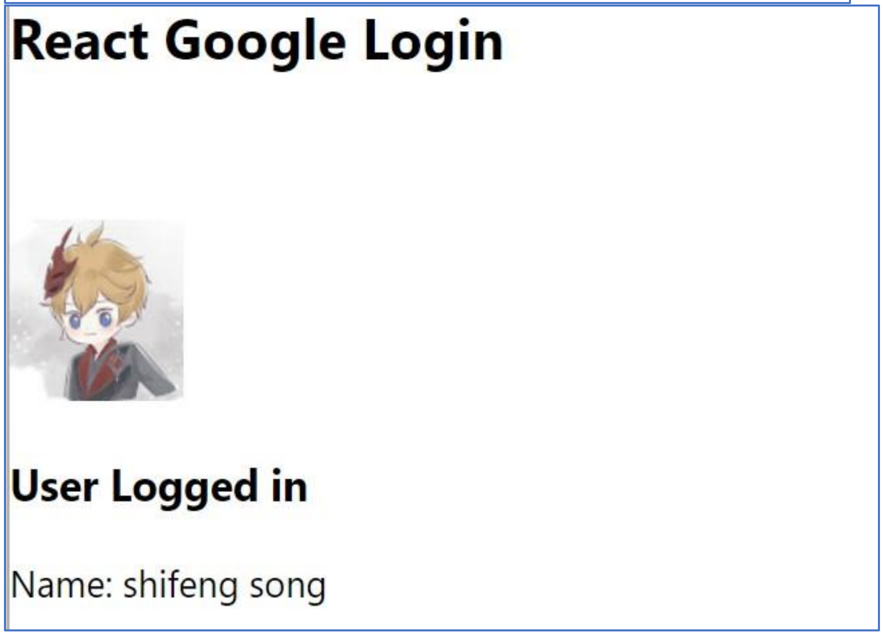
```
PS C:\Users\a2452\Desktop\tmp\8919\assignment 1\front-end\k8s\syslogclient> kubectl port-forward google-sso-785d974f65-q
nmsq 8080:80
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::1]:8080 -> 80
```

9. Go to browser: http://localhost:8080/


localhost:8080

React Google Login

Sign in with Google


React Google Login

User Logged in

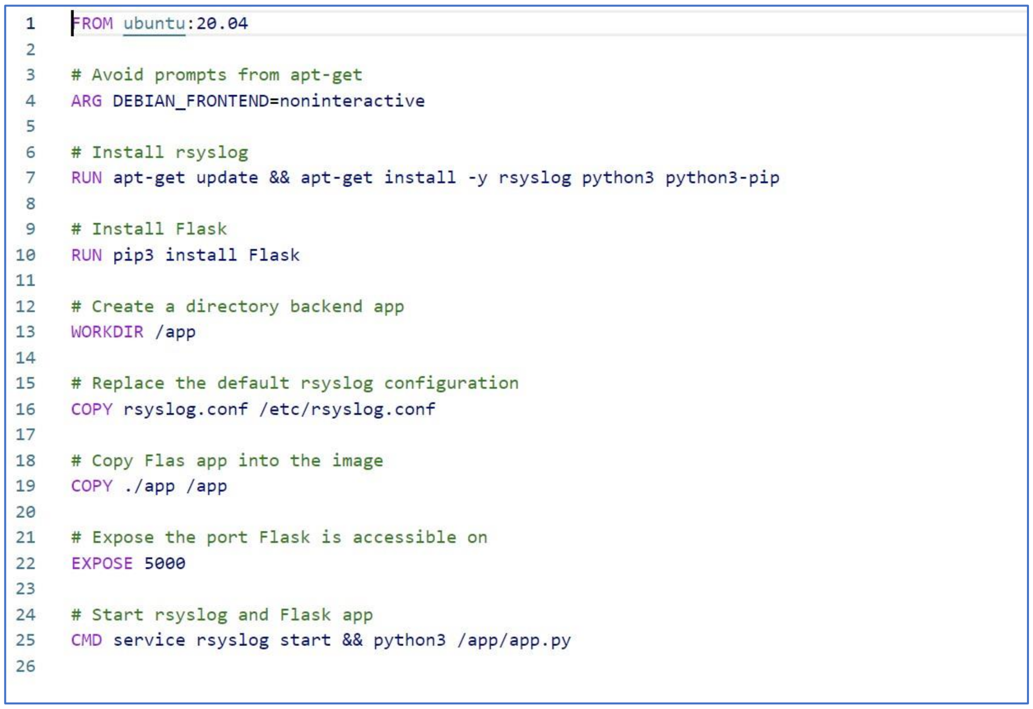
Name: shifeng song

10. success

b. Back-end:

receives logs from front-end, then forwards to syslog server

We used Flask for backend endpoints. To do so the docker file needs to be updated.



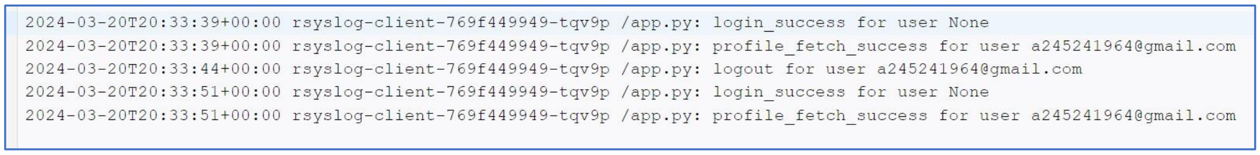
```
FROM ubuntu:20.04

# Avoid prompts from apt-get
ARG DEBIAN_FRONTEND=noninteractive

# Install rsyslog
RUN apt-get update && apt-get install -y rsyslog python3 python3-pip

# Install Flask
RUN pip3 install Flask

# Create a directory backend app
WORKDIR /app

# Replace the default rsyslog configuration
COPY rsyslog.conf /etc/rsyslog.conf

# Copy Flas app into the image
COPY ./app /app

# Expose the port Flask is accessible on
EXPOSE 5000

# Start rsyslog and Flask app
CMD service rsyslog start && python3 /app/app.py

```

Update rsyslog.conf file to monitor /var/log/webapp.log and forward the logs

Create docker image, and push to docker hub

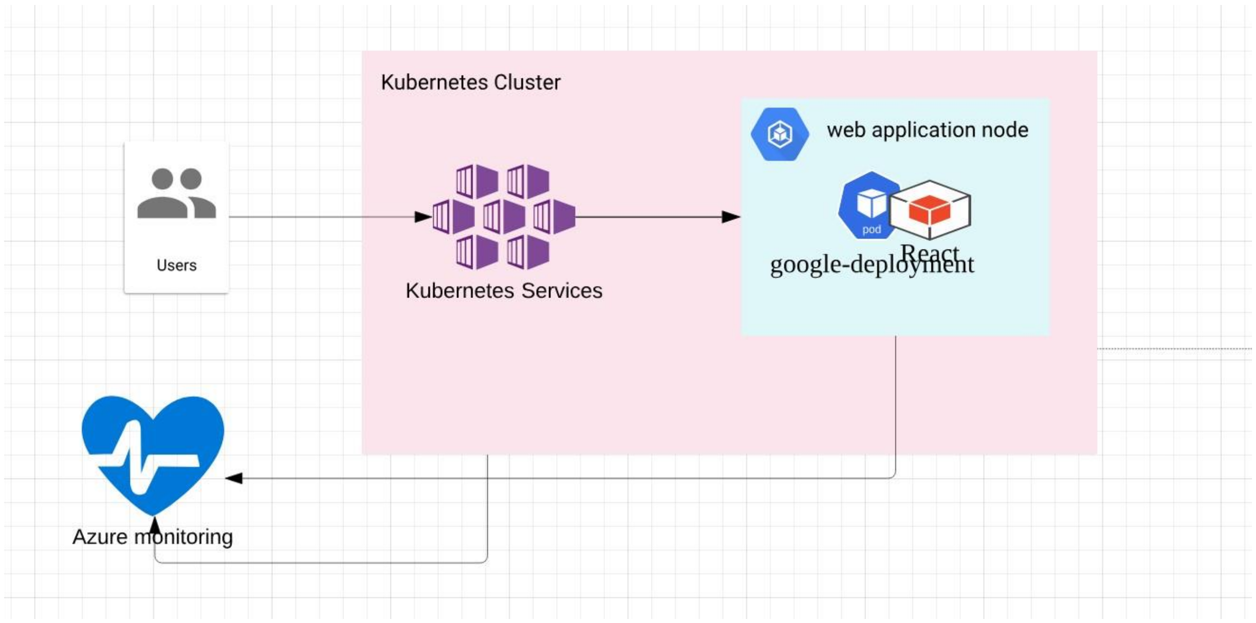
```
docker build -t shifeng1428/rsyslog-
client:latest .
docker push shifeng1428/rsyslog-client
kubectl apply -f rsyslog-client-development.ymal
```

From the rsyslog server
Receives logging informations
Use cat/var/log/rsyslog-client-769f449949 to display logs received from syslog client.



```
2024-03-20T20:33:39+00:00 rsyslog-client-769f449949-tqv9p /app.py: login_success for user None
2024-03-20T20:33:39+00:00 rsyslog-client-769f449949-tqv9p /app.py: profile_fetch_success for user a245241964@gmail.com
2024-03-20T20:33:44+00:00 rsyslog-client-769f449949-tqv9p /app.py: logout for user a245241964@gmail.com
2024-03-20T20:33:51+00:00 rsyslog-client-769f449949-tqv9p /app.py: login_success for user None
2024-03-20T20:33:51+00:00 rsyslog-client-769f449949-tqv9p /app.py: profile_fetch_success for user a245241964@gmail.com
```

Architecture (Phase II)



After the extensive testing and troubleshooting on the local K8S extension with Docker on the local computer, we realized there are too many configuration steps and too many moving parts. Such as the syslog server and client's IP address in AKS cluster could be changed without any notice. After careful research and testing, we figured out more straight-forward way to transfer all syslogs from AKS cluster to Azure Log Analytics. We carefully examined the requirements of the assignment and agreed on the more straight-forward approach.

At the outset of this workflow stand the users, engaging directly with the web application housed in the Kubernetes cluster. This cluster acts as the hosting environment for the React application which benefits from the orchestration and networking capabilities of Kubernetes services. These services are instrumental in facilitating the application's deployment and ensuring seamless interconnectivity.

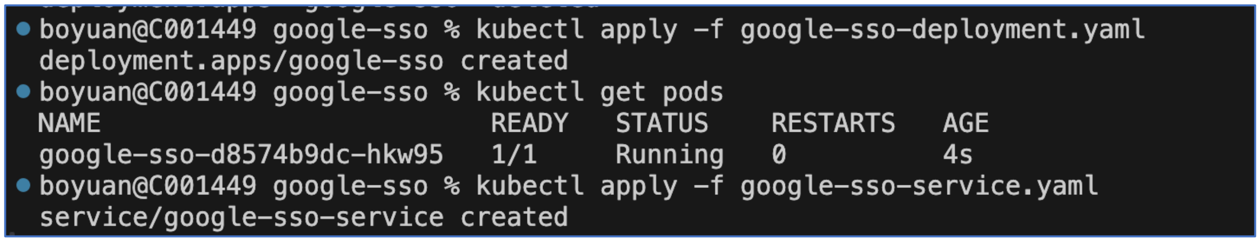
Within this schema, the node hosting the React web application, identified as **google-deployment**, marks a departure from the prior arrangement. Here, the application employs an npm module for the direct transmission of login data to the Azure log database. This refinement streamlines the logging architecture, diminishing the complexity by reducing intermediary processes. Such an approach holds the promise of enhancing the logging mechanism's efficiency and reliability.

Moreover, Azure Monitoring extends its oversight beyond application logs to encompass the syslogs from the virtual machines that underpin the Kubernetes cluster. This direct link fosters an immediate and thorough examination of both performance and system logs across the entire cluster, offering a thorough view of its operational health and activities.

To summarize, this enhanced iteration allows a more direct and efficient logging process. The React web application leverages npm modules for immediate interaction with the Azure log database concerning authentication logs. At the same time, the virtual machines within the Kubernetes framework transmit syslogs directly to Azure Log Monitor, showcasing a more streamlined method in logging and monitoring operations.

Deployments

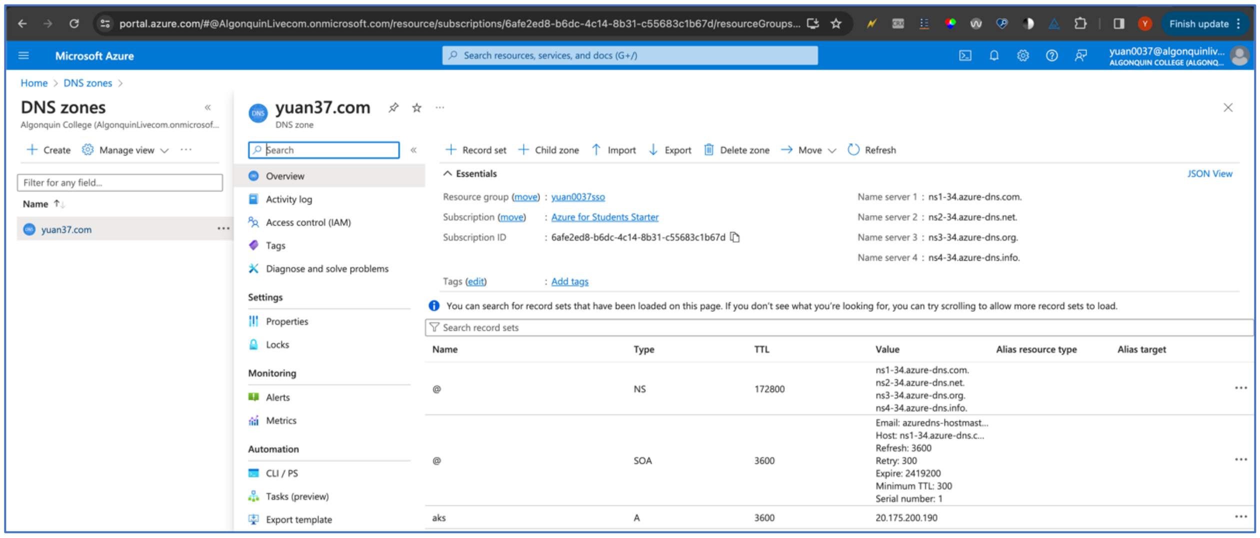
1. Deployment of google-sso



```
● boyuan@C001449 google-sso % kubectl apply -f google-sso-deployment.yaml
  deployment.apps/google-sso created
● boyuan@C001449 google-sso % kubectl get pods
  NAME                          READY   STATUS    RESTARTS   AGE
  google-sso-d8574b9dc-hkw95    1/1     Running   0          4s
● boyuan@C001449 google-sso % kubectl apply -f google-sso-service.yaml
  service/google-sso-service created
```

2. Deployment of a Custom Domain

A custom domain and sub domain with DNS zone were set up with Azure DNS Zones (required for Google SSO's origin and redirection to work properly)



portal.azure.com/#@AlgonquinLivecom.onmicrosoft.com/resource/subscriptions/6afe2ed8-b6dc-4c14-8b31-c55683c1b67d/resourceGroups...

Microsoft Azure — yuan0037@algonquinliv... ALGONQUIN COLLEGE (ALGONQ...

Home > DNS zones >

DNS zones — Algonquin College (AlgonquinLivecom.onmicrosof...

yuan37.com — DNS zone

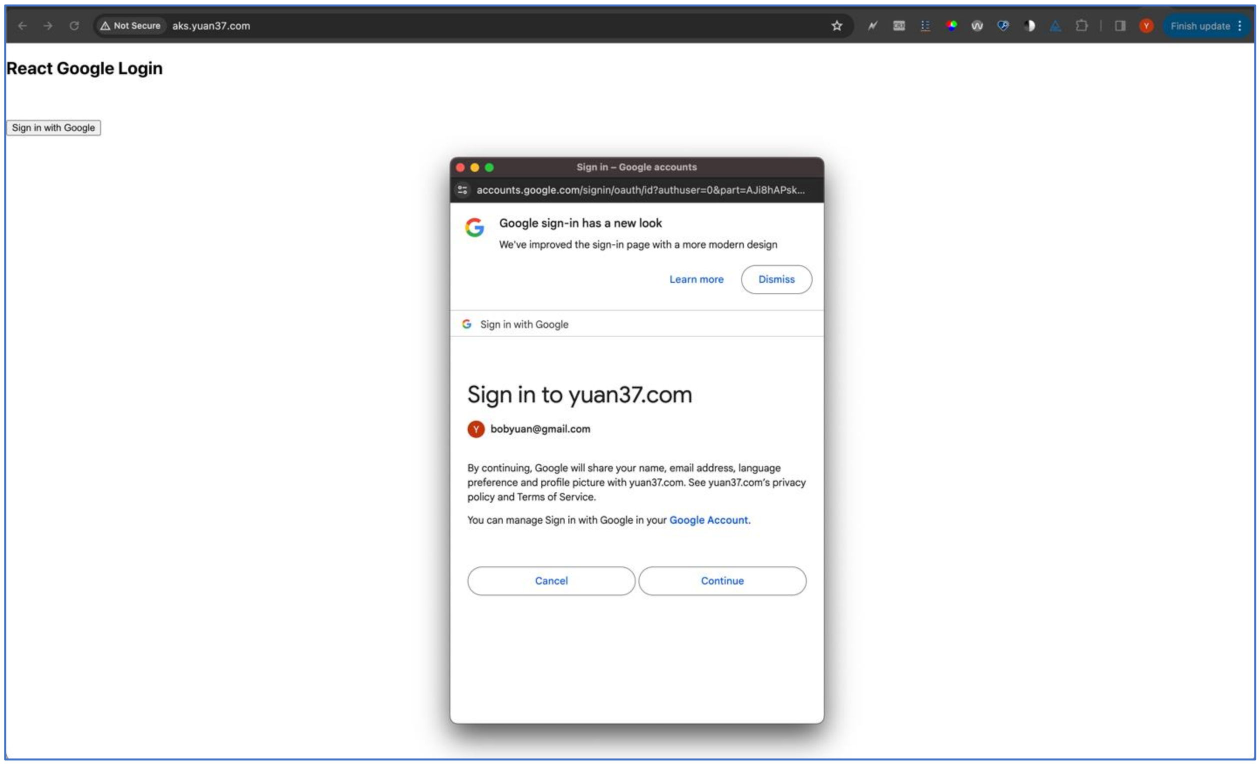
Essentials

Resource group (move)	yuan0037sso
Subscription (move)	Azure for Students Starter
Subscription ID	6afe2ed8-b6dc-4c14-8b31-c55683c1b67d
Tags (edit)	Add tags
Name server 1	ns1-34.azure-dns.com.
Name server 2	ns2-34.azure-dns.net.
Name server 3	ns3-34.azure-dns.org.
Name server 4	ns4-34.azure-dns.info.

You can search for record sets that have been loaded on this page. If you don't see what you're looking for, you can try scrolling to allow more record sets to load.

Name	Type	TTL	Value	Alias resource type	Alias target
@	NS	172800	ns1-34.azure-dns.com. ns2-34.azure-dns.net. ns3-34.azure-dns.org. ns4-34.azure-dns.info.		
@	SOA	3600	Email: azuredns-hostmast... Host: ns1-34.azure-dns.c... Refresh: 3600 Retry: 300 Expire: 2419200 Minimum TTL: 300 Serial number: 1		
aks	A	3600	20.175.200.190		

3. Go to the URL of the sub domain (https://aks.yuan37.com) and sign in.



Not Secure aks.yuan37.com

React Google Login

Sign in – Google accounts

accounts.google.com/signin/oauth/id?authuser=0&part=AJi8hAPsk...

Google sign-in has a new look

We've improved the sign-in page with a more modern design

Sign in with Google

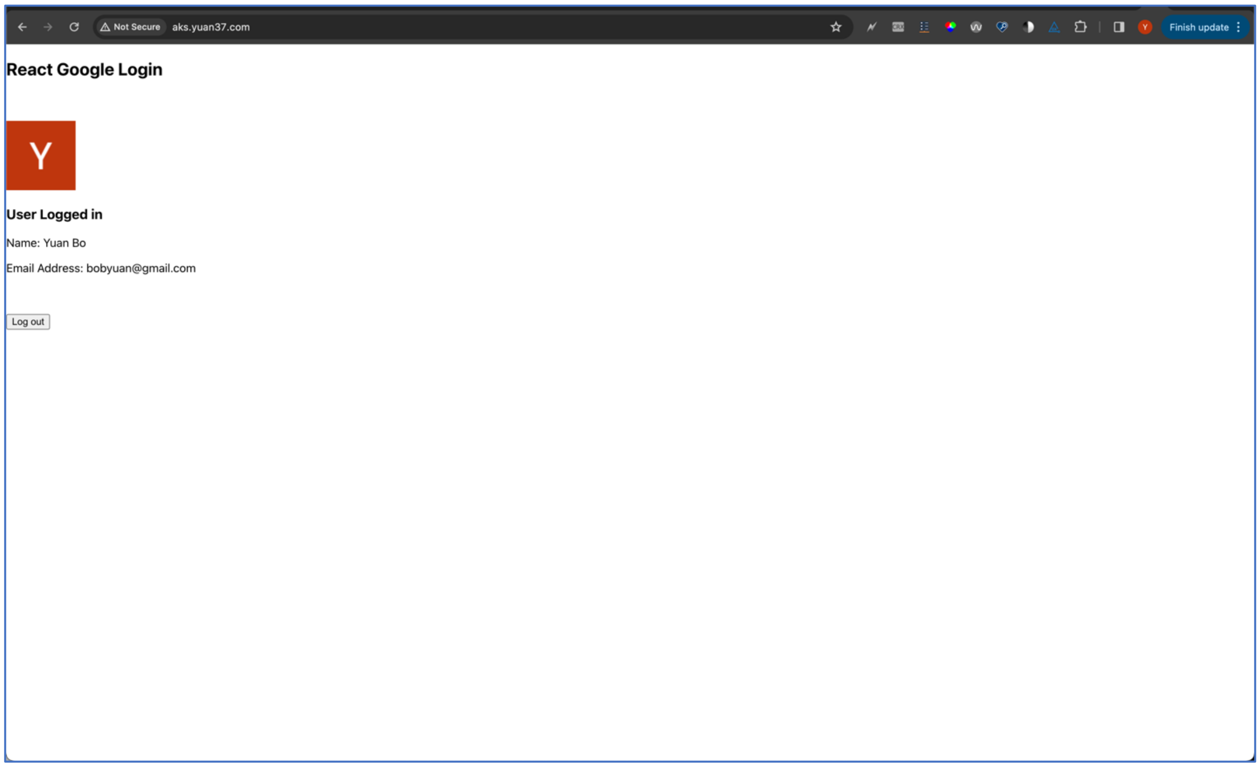
Sign in to yuan37.com

bobyuan@gmail.com

By continuing, Google will share your name, email address, language preference and profile picture with yuan37.com. See yuan37.com's privacy policy and Terms of Service.

You can manage Sign in with Google in your Google Account.

.



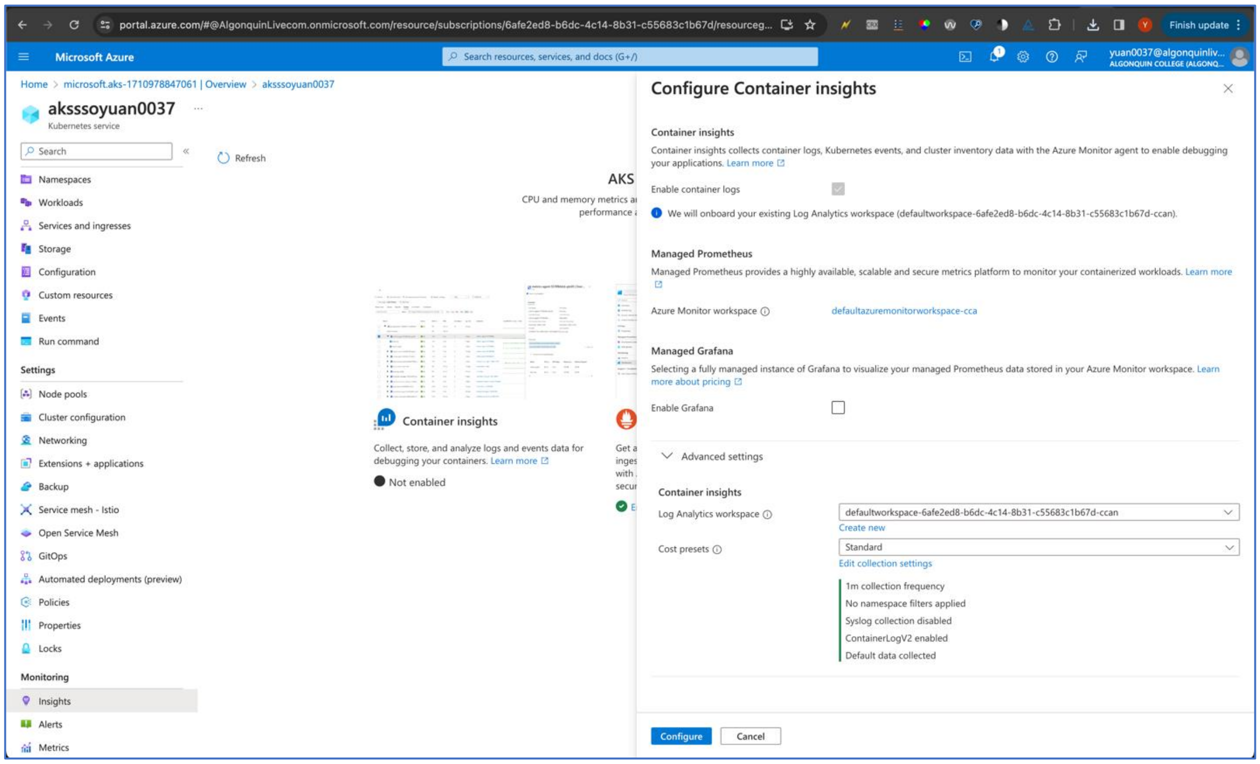
React Google Login

User Logged in

Name: Yuan Bo

Email Address: bobyuan@gmail.com

4. Enable "Insights" for the cluster.



Configure Container insights

Container insights

Container insights collects container logs, Kubernetes events, and cluster inventory data with the Azure Monitor agent to enable debugging your applications. Learn more

Enable container logs ✓

We will onboard your existing Log Analytics workspace (defaultworkspace-6afe2ed8-b6dc-4c14-8b31-c55683c1b67d-ccan).

Managed Prometheus

Managed Prometheus provides a highly available, scalable and secure metrics platform to monitor your containerized workloads. Learn more

Azure Monitor workspace: defaultazuremonitorworkspace-cca

Managed Grafana

Selecting a fully managed instance of Grafana to visualize your managed Prometheus data stored in your Azure Monitor workspace. Learn more about pricing

Enable Grafana ☐

Advanced settings

Container insights

Log Analytics workspace: defaultworkspace-6afe2ed8-b6dc-4c14-8b31-c55683c1b67d-ccan

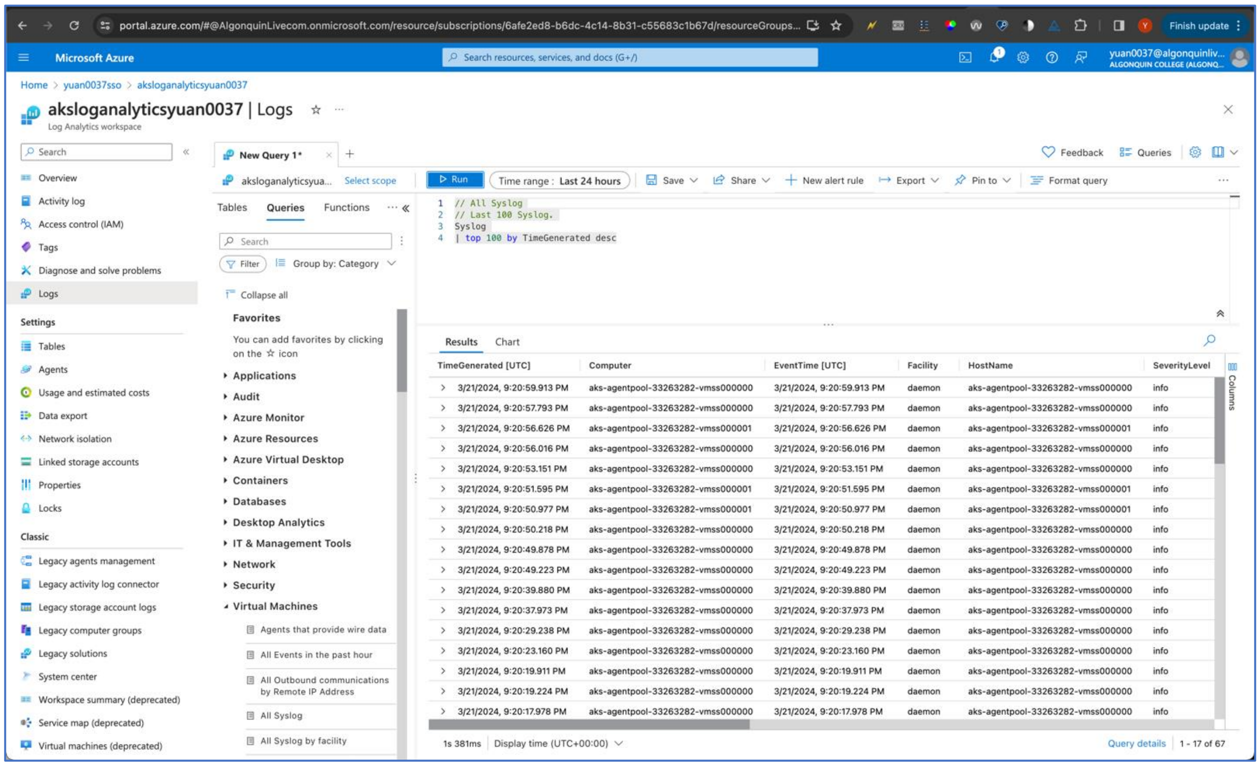
Create new

Cost presets: Standard

Edit collection settings

1m collection frequency
No namespace filters applied
Syslog collection disabled
ContainerLogV2 enabled
Default data collected

5. Verify the syslogs in Azure Log Analytics.



```
// All Syslog
// Last 100 Syslog.
Syslog
| top 100 by TimeGenerated desc
```

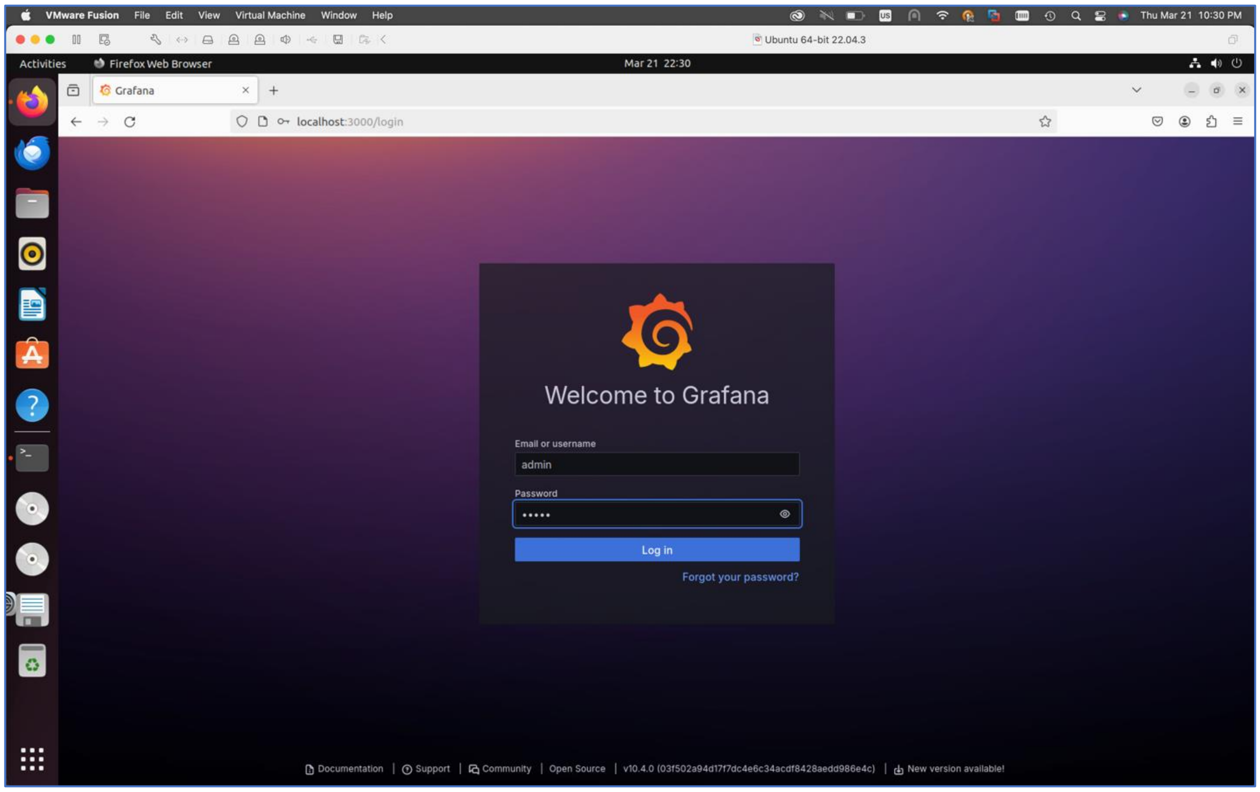
TimeGenerated [UTC]	Computer	EventTime [UTC]	Facility	HostName	SeverityLevel
3/21/2024, 9:20:59.913 PM	aks-agentpool-33263282-vmss000000	3/21/2024, 9:20:59.913 PM	daemon	aks-agentpool-33263282-vmss000000	info
3/21/2024, 9:20:57.793 PM	aks-agentpool-33263282-vmss000000	3/21/2024, 9:20:57.793 PM	daemon	aks-agentpool-33263282-vmss000000	info
3/21/2024, 9:20:56.626 PM	aks-agentpool-33263282-vmss000001	3/21/2024, 9:20:56.626 PM	daemon	aks-agentpool-33263282-vmss000001	info
3/21/2024, 9:20:56.016 PM	aks-agentpool-33263282-vmss000000	3/21/2024, 9:20:56.016 PM	daemon	aks-agentpool-33263282-vmss000000	info
3/21/2024, 9:20:53.151 PM	aks-agentpool-33263282-vmss000000	3/21/2024, 9:20:53.151 PM	daemon	aks-agentpool-33263282-vmss000000	info
3/21/2024, 9:20:51.595 PM	aks-agentpool-33263282-vmss000001	3/21/2024, 9:20:51.595 PM	daemon	aks-agentpool-33263282-vmss000001	info
3/21/2024, 9:20:50.977 PM	aks-agentpool-33263282-vmss000001	3/21/2024, 9:20:50.977 PM	daemon	aks-agentpool-33263282-vmss000001	info
3/21/2024, 9:20:50.218 PM	aks-agentpool-33263282-vmss000000	3/21/2024, 9:20:50.218 PM	daemon	aks-agentpool-33263282-vmss000000	info
3/21/2024, 9:20:49.878 PM	aks-agentpool-33263282-vmss000000	3/21/2024, 9:20:49.878 PM	daemon	aks-agentpool-33263282-vmss000000	info
3/21/2024, 9:20:49.223 PM	aks-agentpool-33263282-vmss000000	3/21/2024, 9:20:49.223 PM	daemon	aks-agentpool-33263282-vmss000000	info
3/21/2024, 9:20:39.880 PM	aks-agentpool-33263282-vmss000000	3/21/2024, 9:20:39.880 PM	daemon	aks-agentpool-33263282-vmss000000	info
3/21/2024, 9:20:37.973 PM	aks-agentpool-33263282-vmss000000	3/21/2024, 9:20:37.973 PM	daemon	aks-agentpool-33263282-vmss000000	info
3/21/2024, 9:20:29.238 PM	aks-agentpool-33263282-vmss000000	3/21/2024, 9:20:29.238 PM	daemon	aks-agentpool-33263282-vmss000000	info
3/21/2024, 9:20:23.160 PM	aks-agentpool-33263282-vmss000000	3/21/2024, 9:20:23.160 PM	daemon	aks-agentpool-33263282-vmss000000	info
3/21/2024, 9:20:19.911 PM	aks-agentpool-33263282-vmss000000	3/21/2024, 9:20:19.911 PM	daemon	aks-agentpool-33263282-vmss000000	info
3/21/2024, 9:20:19.224 PM	aks-agentpool-33263282-vmss000000	3/21/2024, 9:20:19.224 PM	daemon	aks-agentpool-33263282-vmss000000	info
3/21/2024, 9:20:17.978 PM	aks-agentpool-33263282-vmss000000	3/21/2024, 9:20:17.978 PM	daemon	aks-agentpool-33263282-vmss000000	info

1s 381ms | Display time (UTC+00:00) | Query details | 1 - 17 of 67

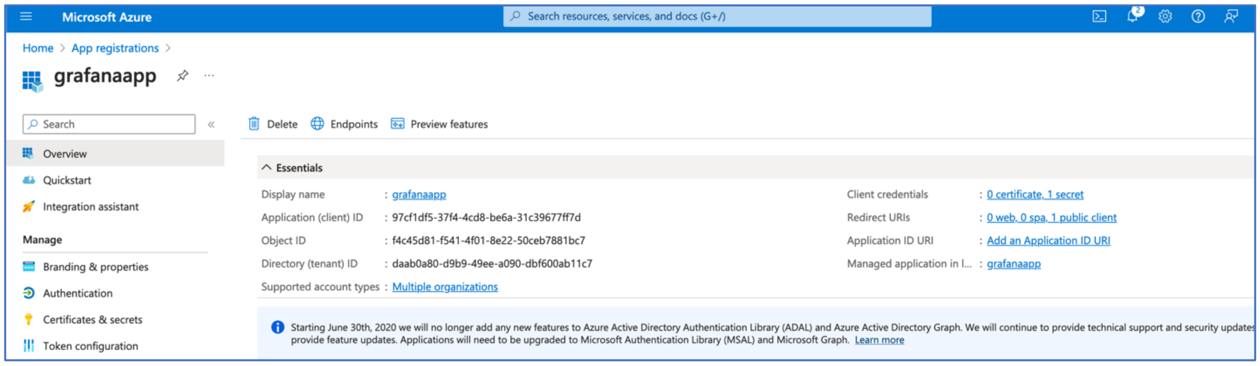
6. Done

Grafana

1. Installation according to the instructions at https://grafana.com/docs/grafana/latest/setup-grafana/installation/debian/

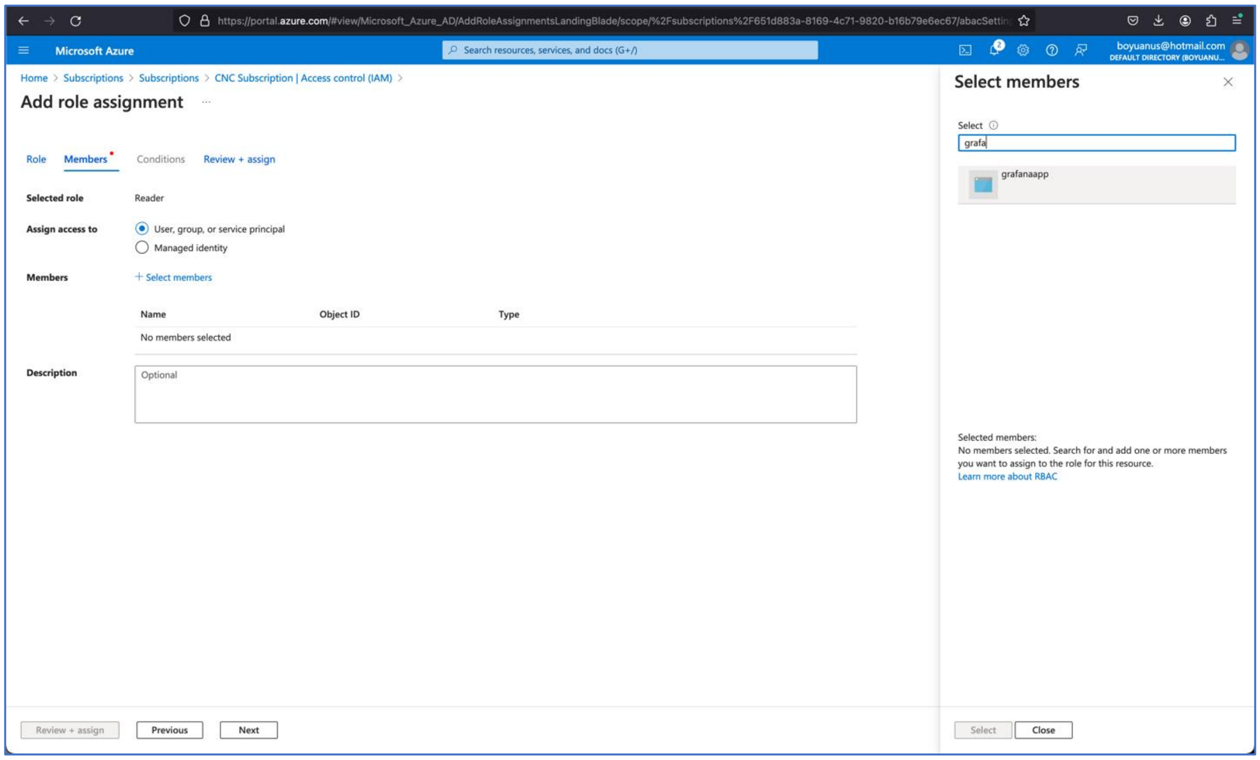

VMware Fusion File Edit View Virtual Machine Window Help
Thu Mar 21 10:30 PM
Ubuntu 64-bit 22.04.3
Activities
Firefox Web Browser
Mar 21 22:30
Grafana
localhost:3000/login
Welcome to Grafana
Email or username
admin
Password
Log in
Forgot your password?
Documentation | Support | Community | Open Source | v10.4.0 (03f502a94d17f7dc4e6c34acdf8428aedd986e4c) | New version available!

2. Create a new “App Registration” in azure called “grafana” and add the reader role assignment to the new app registered.



Display name : grafanaapp
Application (client) ID : 97cf1df5-37f4-4cd8-be6a-31c39677ff7d
Object ID : f4c45d81-f541-4f01-8e22-50ceb7881bc7
Directory (tenant) ID : daab0a80-d9b9-49ee-a090-dbf600ab11c7
Supported account types : Multiple organizations
Client credentials : 0 certificate, 1 secret
Redirect URIs : 0 web, 0 spa, 1 public client
Application ID URI : Add an Application ID URI
Managed application in l... : grafanaapp

Starting June 30th, 2020 we will no longer add any new features to Azure Active Directory Authentication Library (ADAL) and Azure Active Directory Graph. We will continue to provide technical support and security updates provide feature updates. Applications will need to be upgraded to Microsoft Authentication Library (MSAL) and Microsoft Graph. Learn more



https://portal.azure.com/#view/Microsoft_Azure_AD/AddRoleAssignmentsLandingBlade/scope/%2Fsubscriptions%2F651d883a-8169-4c71-9820-b16b79e6ec67/abacSettin

Home > Subscriptions > Subscriptions > CNC Subscription | Access control (IAM) >

Add role assignment

Role | Members | Conditions | Review + assign

Selected role	Reader
Assign access to	(●) User, group, or service principal ( ) Managed identity
Members	+ Select members

Name	Object ID	Type
No members selected		

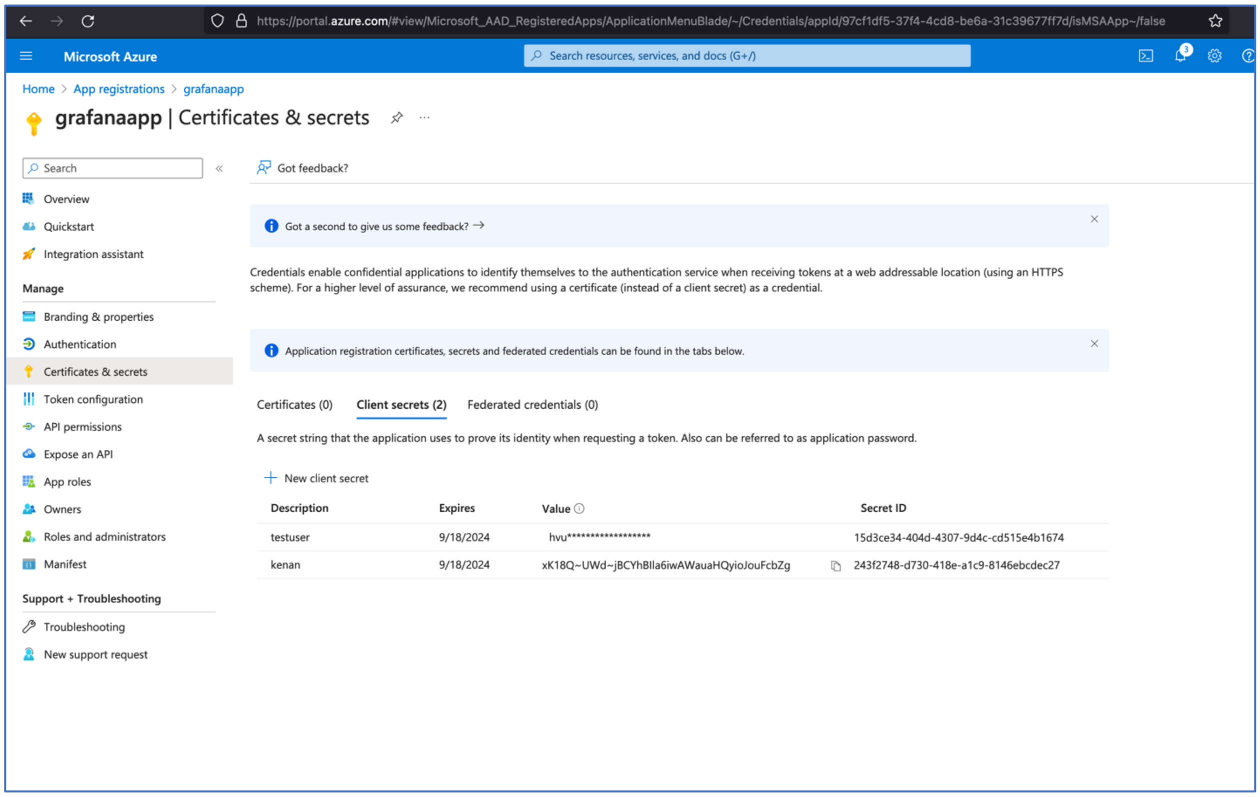
Description: Optional

Select members

Select: grafa

grafanaapp

Selected members:
No members selected. Search for and add one or more members you want to assign to the role for this resource.
Learn more about RBAC



https://portal.azure.com/#view/Microsoft_AAD_RegisteredApps/ApplicationMenuBlade/~/Credentials/appId/97cf1df5-37f4-4cd8-be6a-31c39677ff7d/isMSAApp~/false

Home > App registrations > grafanaapp

grafanaapp | Certificates & secrets

Got a second to give us some feedback? →

Credentials enable confidential applications to identify themselves to the authentication service when receiving tokens at a web addressable location (using an HTTPS scheme). For a higher level of assurance, we recommend using a certificate (instead of a client secret) as a credential.

Application registration certificates, secrets and federated credentials can be found in the tabs below.

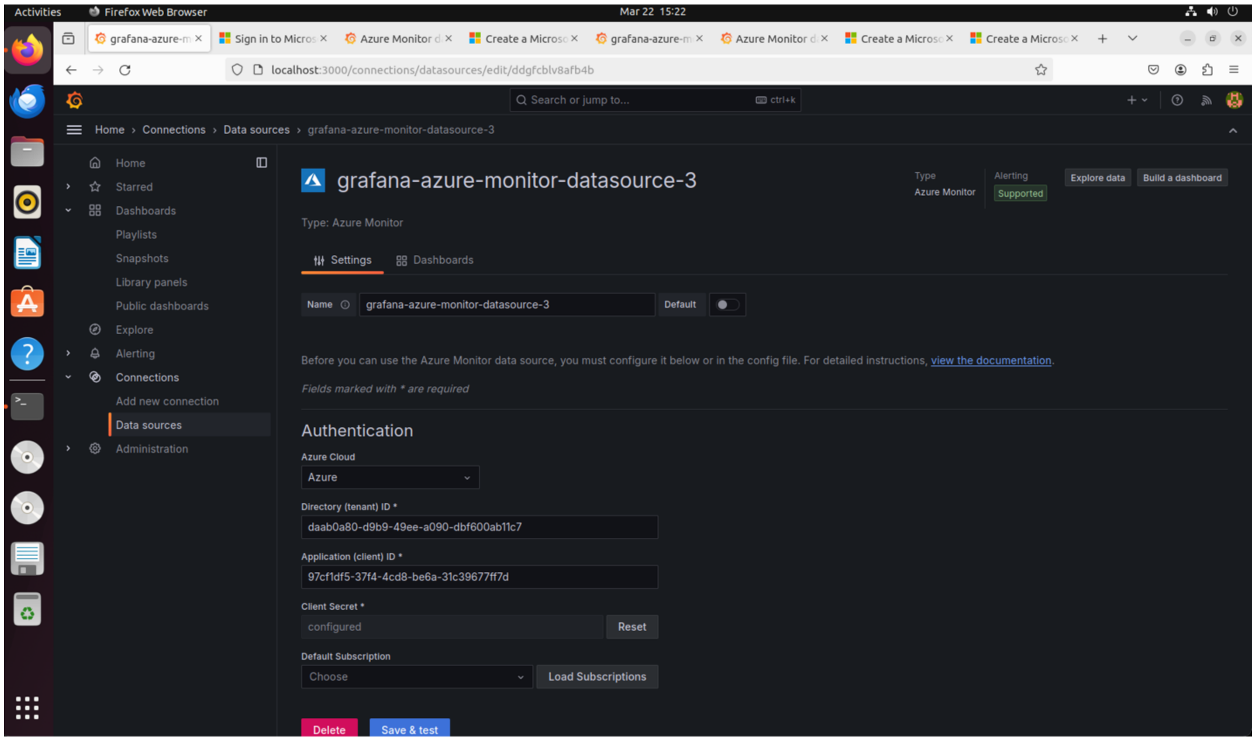
Certificates (0) | Client secrets (2) | Federated credentials (0)

A secret string that the application uses to prove its identity when requesting a token. Also can be referred to as application password.

+ New client secret

Description	Expires	Value	Secret ID
testuser	9/18/2024	hvu******************	15d3ce34-404d-4307-9d4c-cd515e4b1674
kenan	9/18/2024	xK18Q~UWd~jBCYhBlla6iwAWauaHQyioJouFcbZg	243f2748-d730-418e-a1c9-8146ebcdec27

3. Use the credentials and settings of the “grafana” app added in Azure to set up the datasource in Grafana



localhost:3000/connections/datasources/edit/ddgfcblv8afb4b

Home › Connections › Data sources › grafana-azure-monitor-datasource-3

grafana-azure-monitor-datasource-3

Type: Azure Monitor

Name: grafana-azure-monitor-datasource-3

Before you can use the Azure Monitor data source, you must configure it below or in the config file. For detailed instructions, view the documentation.

*Fields marked with * are required*

Authentication

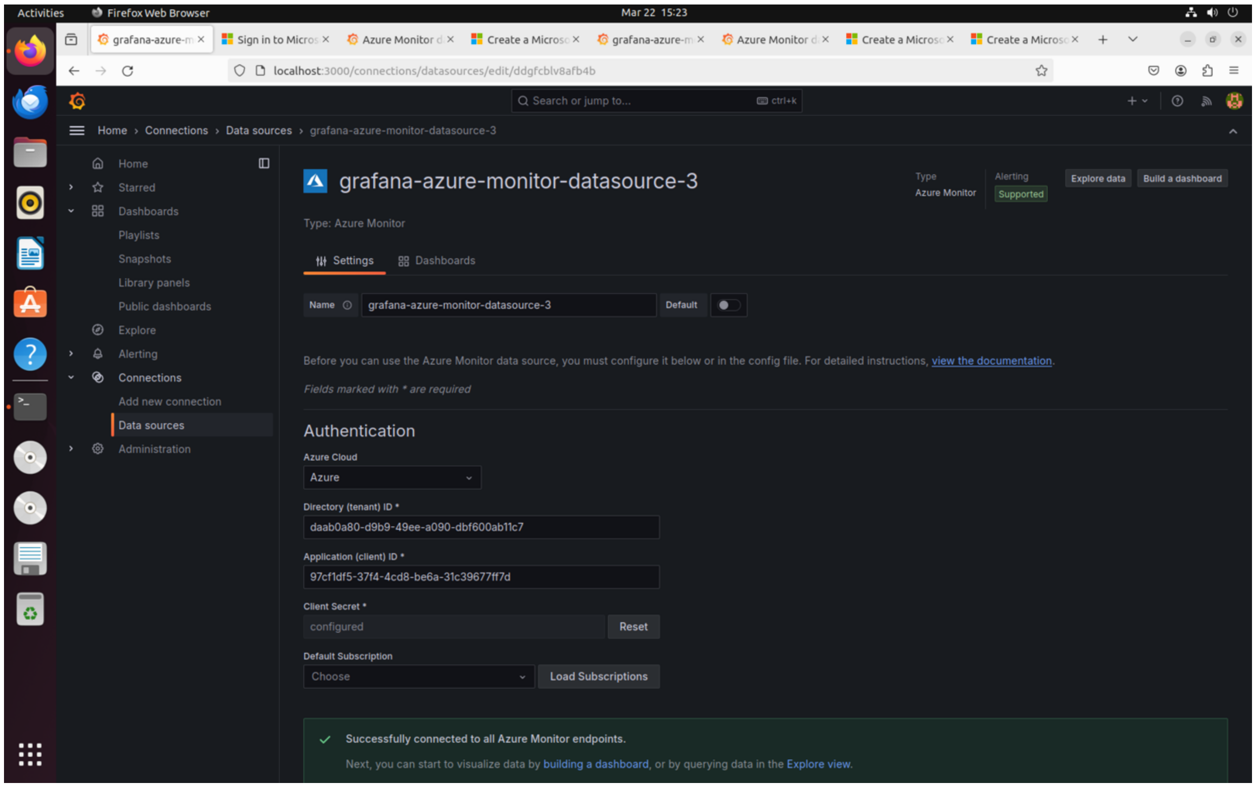
Azure Cloud: Azure

Directory (tenant) ID *: daab0a80-d9b9-49ee-a090-dbf600ab11c7

Application (client) ID *: 97cf1df5-37f4-4cd8-be6a-31c39677ff7d

Client Secret *: configured

Default Subscription: Choose



localhost:3000/connections/datasources/edit/ddgfcblv8afb4b

Home › Connections › Data sources › grafana-azure-monitor-datasource-3

grafana-azure-monitor-datasource-3

Type: Azure Monitor

Name: grafana-azure-monitor-datasource-3

Before you can use the Azure Monitor data source, you must configure it below or in the config file. For detailed instructions, view the documentation.

*Fields marked with * are required*

Authentication

Azure Cloud: Azure

Directory (tenant) ID *: daab0a80-d9b9-49ee-a090-dbf600ab11c7

Application (client) ID *: 97cf1df5-37f4-4cd8-be6a-31c39677ff7d

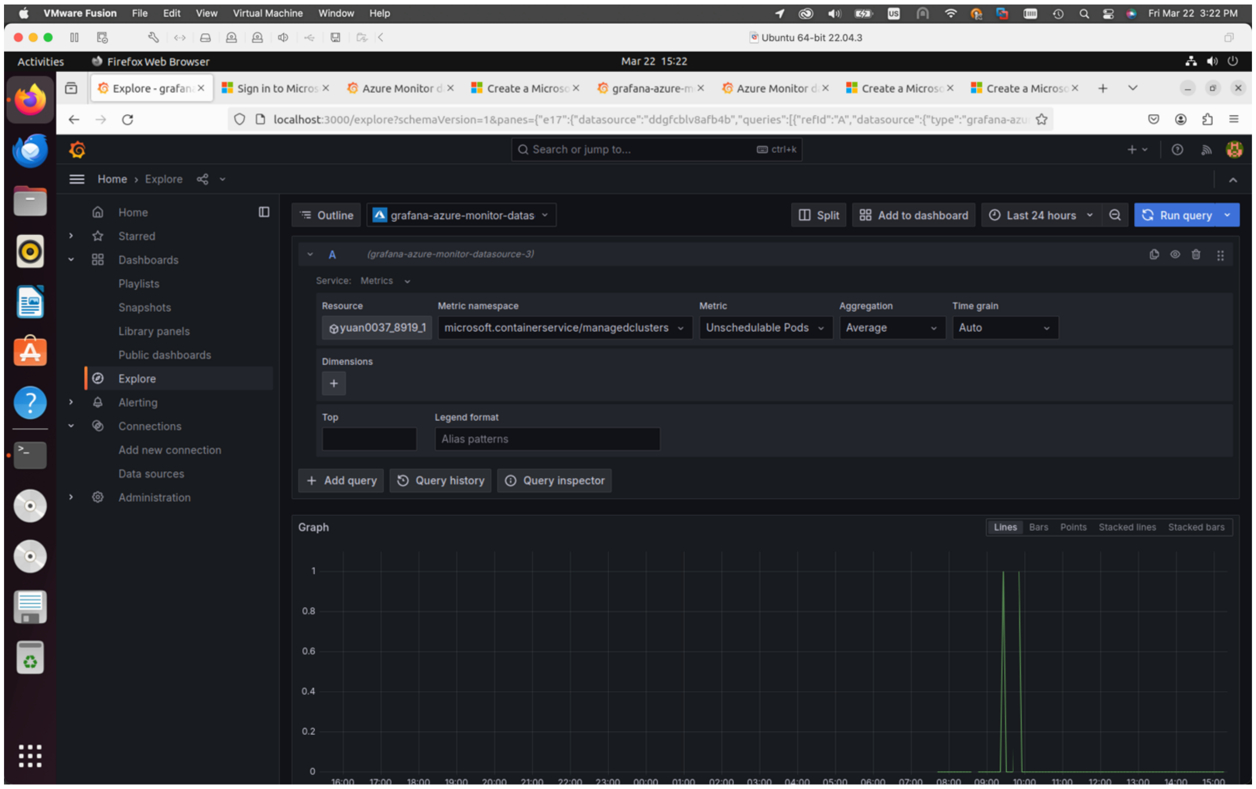
Client Secret *: configured

Default Subscription: Choose

Successfully connected to all Azure Monitor endpoints.

Next, you can start to visualize data by building a dashboard, or by querying data in the Explore view.

4. Create a dashboard and get live data.


localhost:3000/explore?schemaVersion=1&panes={"e17":{"datasource":"ddgfcblv8afb4b","queries":[{"refId":"A","datasource":{"type":"grafana-azu

grafana-azure-monitor-datas

(grafana-azure-monitor-datasource-3)

Service: Metrics

Resource: yuan0037_8919_1
Metric namespace: microsoft.containerservice/managedclusters
Metric: Unschedulable Pods
Aggregation: Average
Time grain: Auto

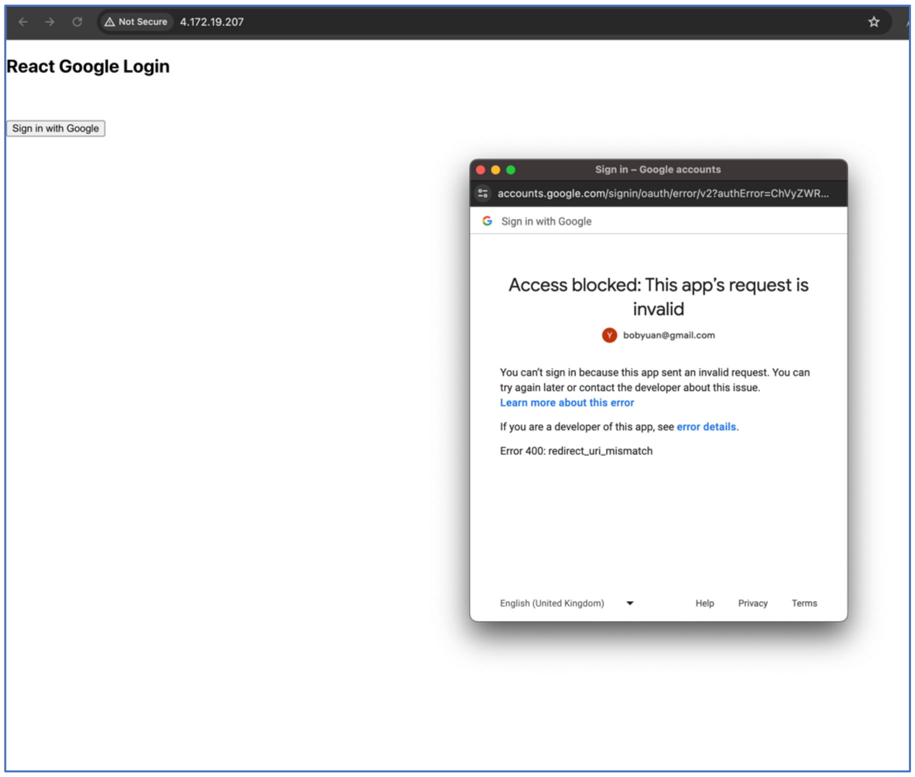
Dimensions

Top

Legend format: Alias patterns

Conclusion

1. A custom domain is required to host a Google SSO app on AKS cluster because Google will automatically verify the origin and redirection URL when the sign on request is received. If it does not match, you will see similar error message as following.



React Google Login

Access blocked: This app's request is invalid

bobyuan@gmail.com

You can't sign in because this app sent an invalid request. You can try again later or contact the developer about this issue.
Learn more about this error

If you are a developer of this app, see error details.

Error 400: redirect_uri_mismatch

2. A self-hosted or on-prem Grafana can only connect to Azure Log Analytics using a set of credentials of a registered app in Azure subscription where the Azure Log Analytics are hosted. To register the app, you need to have "Application Administrator" role. Once the app is registered in Azure via "App Registration", the app needs to be assigned with "Reader" role for Grafana to connect through it to your Log Analytics.

3. AKS cluster's syslog can be seamlessly transmitted to Azure Log Analytics through "Insights" without any intermediary syslog servers.

4. We switched to different subscriptions and provisioned different sets of AKS cluster because of the blockers we experienced below. In the end, we were able to successfully connect Grafana to Azure Log Analytics and consume/render the chart from all kinds of metrics data we got from the AKS cluster that hosts the Google SSO app.

    a. No custom domain name to support deployed SSO service on AKS.
    b. No "Application Administrator" role for our account under Algonquin directory.
    c. Not able to add "reader" role on the application we created that will be used by Grafana to connect to Azure.

References

Bwren. (n.d.). *Syslog collection with container insights - azure monitor*. Azure Monitor | Microsoft Learn. https://learn.microsoft.com/en-us/azure/azure-monitor/containers/container-insights-syslog

ComputingPost. (2022, October 15). *Configure rsyslog centralized log server on ubuntu 22.04: 20.04: 18.04*. Medium. https://computingpost.medium.com/configure-rsyslogcentralized-log-server-on-ubuntu-22-04-20-04-18-04-b5222129b3f3

Install grafana on debian or ubuntu: Grafana documentation. Grafana Labs. (n.d.). https://grafana.com/docs/grafana/latest/setup-grafana/installation/debian/