

Create a workspace in Amazon Grafana

- Go to the [Amazon Grafana](#) console, click on **Create workspace**
- Key in the workspace name **emr-on-tfc-summit**, then **Next**
- Use **AWS SSO** as the authentication access type.


Configure settings [Info](#)

Authentication access [Info](#)

Choose at least one authentication method.

☒ AWS Single Sign-On (AWS SSO)  **Not enabled**

You can enable SSO by creating a user. This new user does not automatically have access to the Grafana console. You will still need to assign this user later, once this workspace is created.

 For SSO to be usable you need to add a user to the service. Note that this new user does not automatically have access to the Grafana console. You will still need to assign this user later, once this workspace is created.

Create user

☐ Security Assertion Markup Language (SAML)

You will need to complete additional steps to finish SAML configuration once this workspace is created.

- Create an SSO user with random username and email, if the SSO is not enabled. It's OK to use an existing cross-region SSO user.

Create user ×

Create a new AWS user. AWS Single Sign-On will be enabled upon creation.

Email address

mysso@companyone.com

First name

my

Last name

SSO

Cancel


Create user

- In the Data sources and notification channels – optional section, select the **Amazon Managed Service for Prometheus**.

▼ Data sources and notification channels - optional

Data sources

Selecting an AWS data source below creates an IAM role that enables Amazon Grafana access to those resources in your current account. It does not set up the selected service as a data source. Note that some resources must be tagged GrafanaDataSource to be accessible.

	Data source name
<input type="checkbox"/>	AWS IoT SiteWise
<input type="checkbox"/>	AWS X-Ray
<input type="checkbox"/>	Amazon CloudWatch
<input type="checkbox"/>	Amazon OpenSearch Service
<input checked="" type="checkbox"/>	Amazon Managed Service for Prometheus
<input type="checkbox"/>	Amazon TimeStream
<input type="checkbox"/>	Amazon Redshift
<input type="checkbox"/>	Amazon Athena

- Proceed to the final Review and create page, then **Create workspace**
- click on the **Assign new user or group** button when you see the warning message in the Authentication tab.

emr-on-tfc-summit

Summary [Info](#)

Description 

-

Grafana workspace URL

g-c8353b2fe0.grafana-workspace.us-east-1.amazonaws.com 

Status

 Active

Date created

2022-04-22

Authentication access

AWS SSO

Authentication

Data sources

Notification channels

Tags

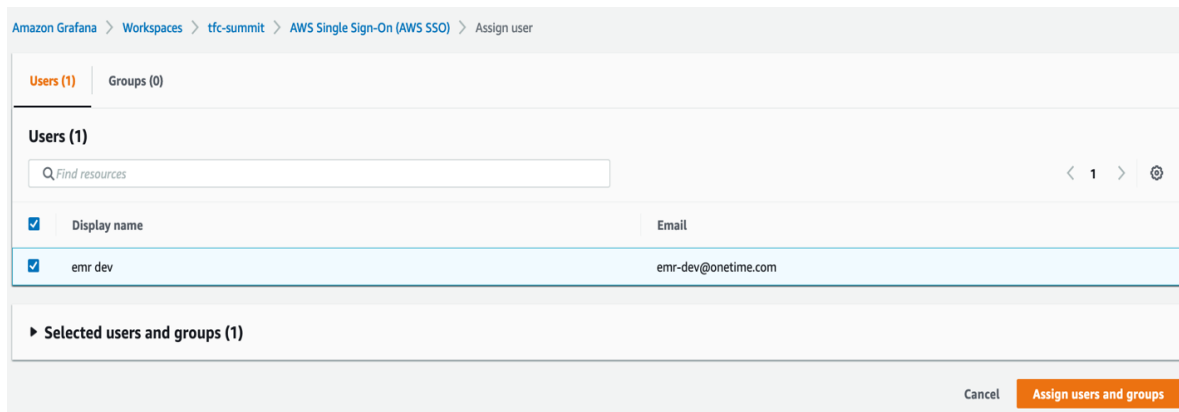
AWS Single Sign-On (SSO)

You can enable AWS SSO by creating a user or connect AWS SSO to an external identity provider (IdP) to enable users to log in to the workspace. If you create a new user, you will need to assign this user access to the workspace before they can log in to the workspace.

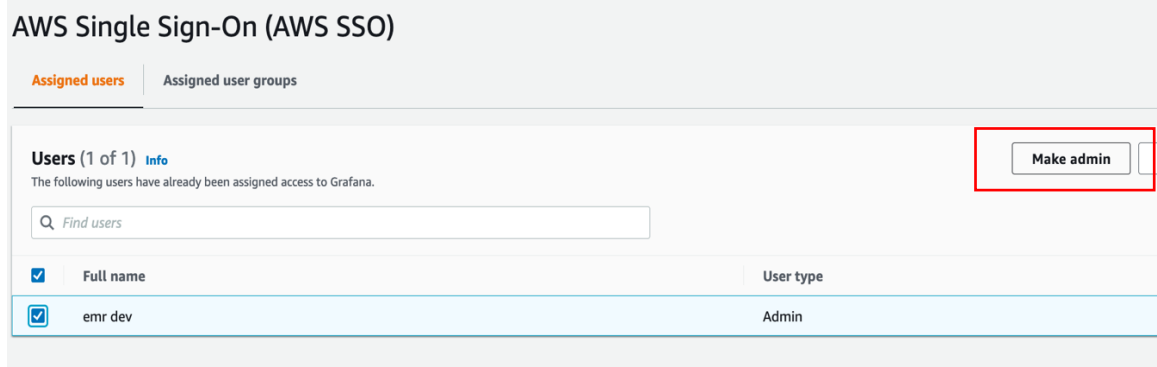
[Assign new user or group](#)

 Assign new users to the Grafana workspace so users can access the workspace URL.

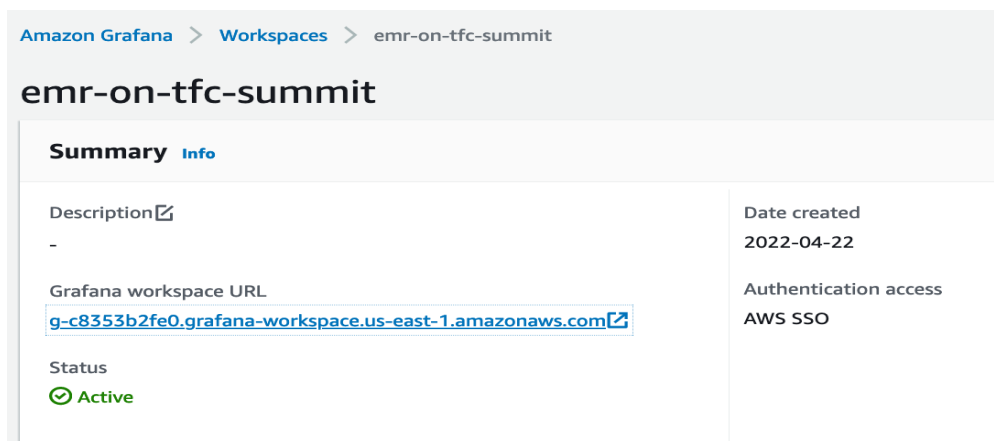
- Assign an SSO user.



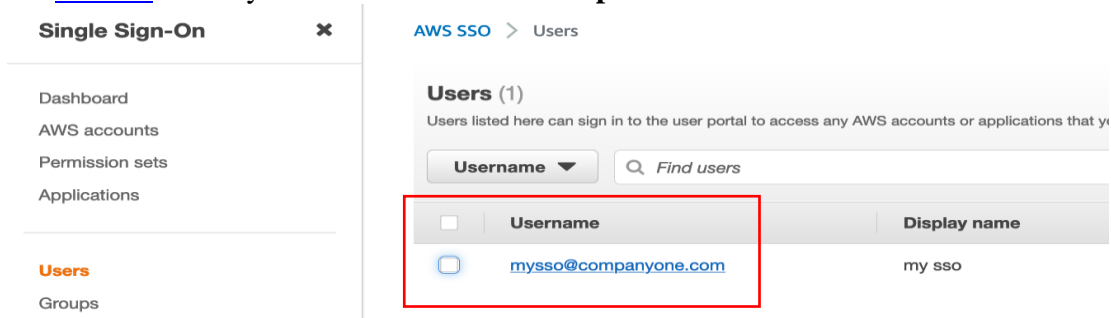
- Re-select the user and **set it as an admin**. This option lets the user add data sources to the Grafana dashboard in the next steps.



- Go back to the workspace console and click the **Grafana workspace URL**. Login via AWS SSO.



- If you don't have the username or password, can get them from the [AWS SSO User console](#). Click your Username then **Reset password**.



- Generate a one-time password

Reset password for user "mysso@companyone.com"

How to reset this user's password?

☐ Send an email to the user with instructions to reset the password
☒ Generate a one-time password and share the password with the user

[Cancel](#)
[Reset password](#)

✔ User password has been reset for user "mysso@companyone.com".

You can copy and share the instructions for signing in to the user portal with the user, or email them the instructions. This is the only time you can view and copy this password.

User portal URL

🔗 <https://d-90674947c4.awsapps.com/start>

Username

🔗 [mysso@companyone.com](#)

One-time password

🔗 [A1Fkgc>W4@&.Z-vr79g^t.&O>6rS4wDoLWRS-zEkqmg&BReEk3\)&tFwD5wO](#)

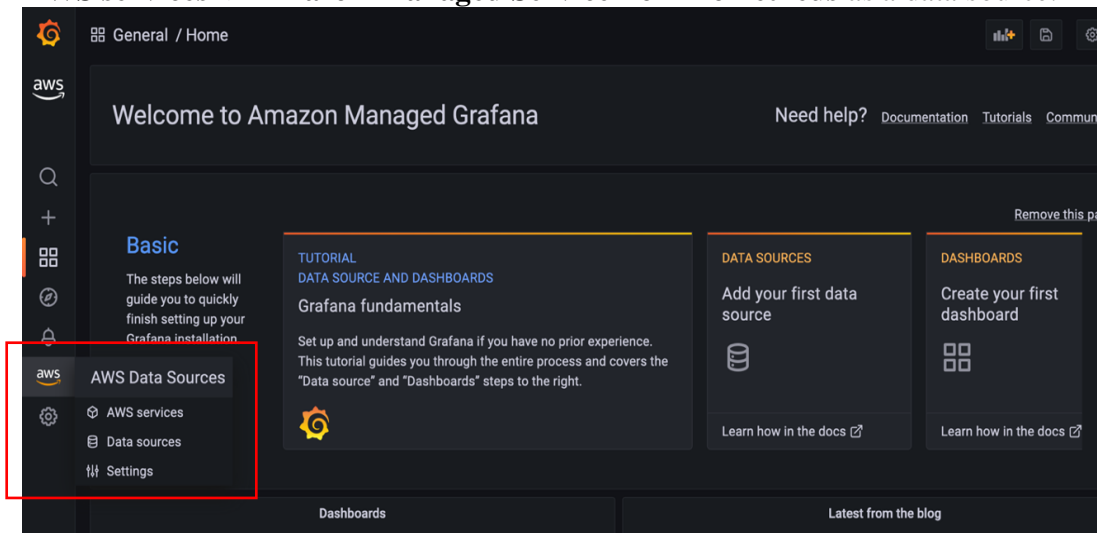
☒ Hide password

[Copy](#)

[Close](#)

Add Prometheus as data source

- After login successfully, select the smaller **AWS logo** on the left ribbon, then choose **AWS services -> Amazon Managed Service for Prometheus** as a data source.



- Choose your **region** and the **data source**, then click **Add 1 data source**. Select the data source with a correct Resource alias, if you have multiple Prometheus data sources.

[AWS services](#)
[Data sources](#)
[Settings](#)

Service Amazon Managed Service for Prometheus

Browse and provision data sources

Specify the required configuration parameters to add data sources.

Regions

US East (N. Virginia)

Region	Resource id	Resource alias
<input checked="" type="checkbox"/> us-east-1	ws-ded1a415-8d45-4de7-9a9e-169107c3f365	tfc-summit

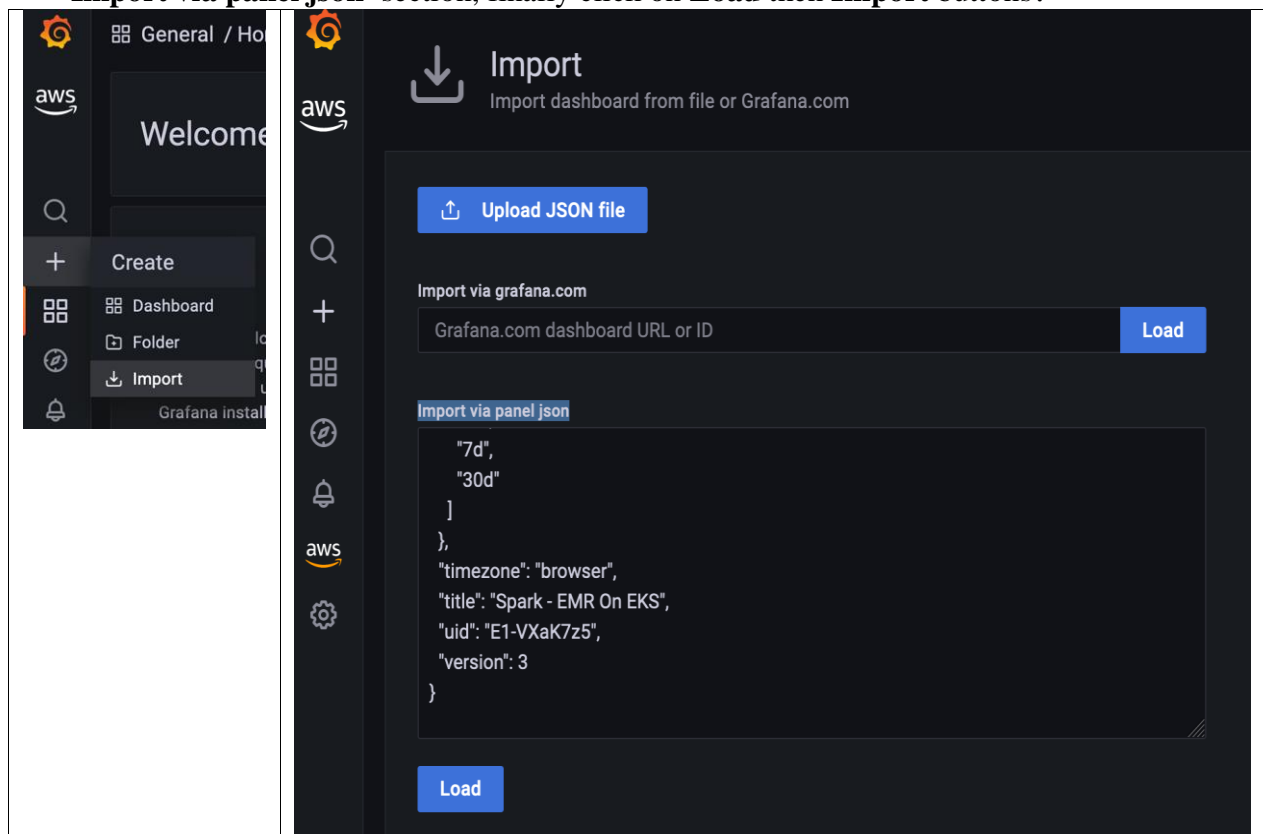
[Add 1 data source](#)

Create a dashboard for Spark

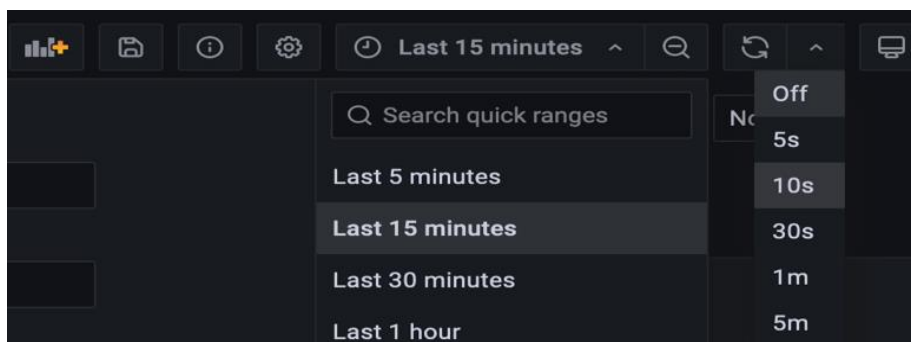
- A pre-defined Spark dashboard template is created already. Open the following link and copy the file content.

<https://raw.githubusercontent.com/aws-ia/terraform-aws-eks-blueprints/main/examples/analytics/emr-on-eks/examples/grafana-dashboard-for-spark/emr-eks-grafana-dashboard.json>

- Click the **+** icon and choose the **Import** option. Paste the template file content to the **Import via panel json** section, finally click on **Load** then **Import** buttons.



- On the dashboard, set the time range to **15 minutes** and change the refresh frequency to **10 seconds**.



- Congratulations! You have successfully setup a Grafana dashboard for EMR on EKS.

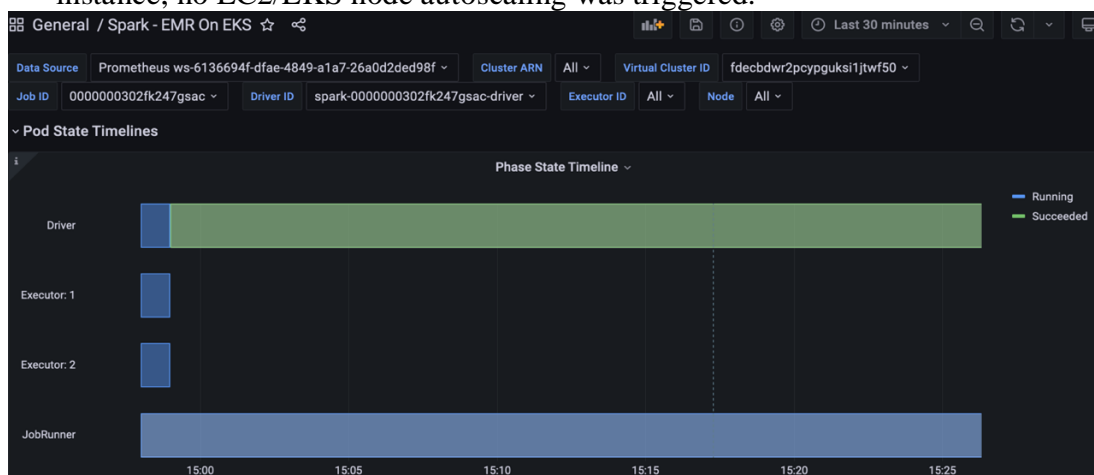
Follow the [workshop instruction](#) to submit a Spark job and monitoring its autoscaling performance on Grafana.

Appendix

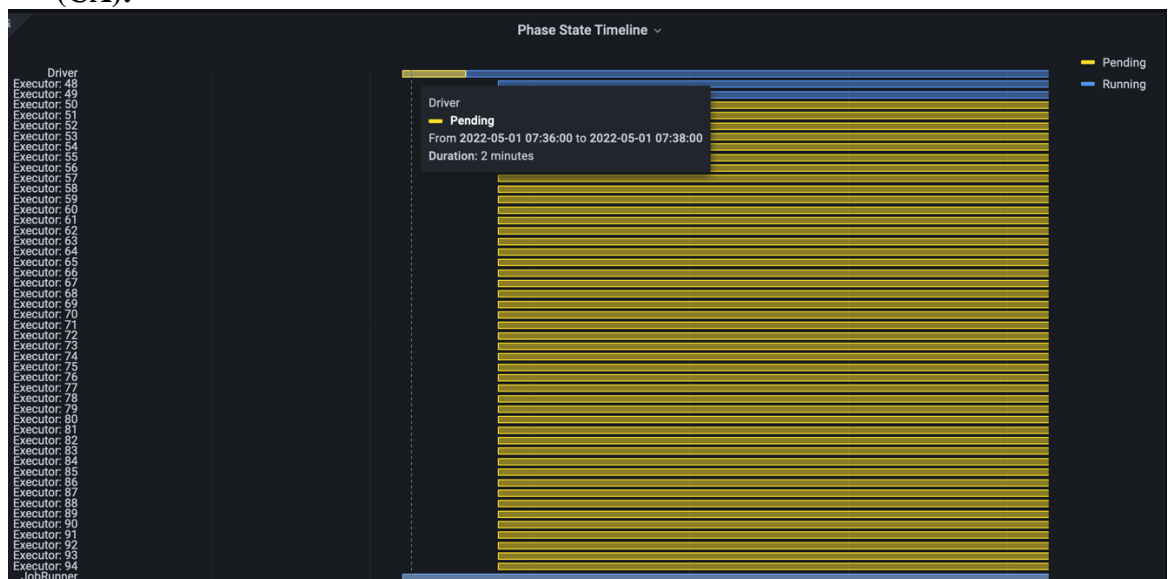
- Understand the dashboard

Pod State Timelines section - A graph that tracks a job autoscaling performance when firing up a Spark application with EMR on EKS. It collects the time & pod status information, then visualize it. It displays when a pod status is changed from pending to running, from running to succeeded.

- The following example shows a 2-executor job was run on an existing EC2 instance, no EC2/EKS node autoscaling was triggered.



- The following case is for a 47-executor job - a medium size Spark application that was still waiting for the compute resources. Because it took approx.3 minutes before starting to scheduling all the executors at once. Unfortunately, we have reached the max number of instance quota after the 3-minute startup time. The job was not running at all. **The autoscaling was managed by Cluster Autoscaler (CA).**



- On the other hand, the same Spark job was submitted to the AZ at the same time, where nodes are scheduled **by Karpenter**. We can see the job was managed to run, simply because of the instant scale-up reaction within 1 minute. Though some pods are still waiting for the available compute resources, over 50% pods were managed to get to the running state.

