

P3 k8s architecture

🧳 Prerequisites

Use terraform to set up one Jenkins servers for code deployment, and one SonarQube Server for code testing on EC2 instances.

For the Jenkins Server, we need to install Jenkins, trivy, docker, and aws cli.

For SonarQube Server, we need to install docker and docker-compose.



Techscrum-Server-Terraform.git
Jamessihang



Connect to GitHub to update

Front end Pipeline

Our code wrote in NodeJs and hosted in an AWS S3 bucket, to integrate with Jenkins, we need to install the necessary plugins.

Go to Manage Plugins in Jenkins, go to available, search and install NodeJS, AWS steps, and bitbucket.

The screenshot shows the Jenkins Manage Plugins interface. A single plugin, "NodeJS 1.5.1", is listed under the "Available" section. It is checked for installation. The plugin details show it was released 10 months ago and is compatible with Jenkins. A note below states: "NodeJS Plugin executes NodeJS script as a build step."

Plugins

The screenshot shows the Jenkins Manage Plugins interface with two plugins installed: "Pipeline: AWS Steps 1.43" and "Bitbucket". Both were installed by the user. The "Pipeline: AWS Steps" plugin is checked for installation. The "Bitbucket" plugin is also checked. Both are released and compatible with Jenkins. A note for the AWS Steps plugin states: "This plugin adds Jenkins pipeline steps to interact with the AWS API".

To integrate Jenkins with Bitbucket we need to generate an app password as following settings.

Personal settings

GENERAL

- Account settings
- Email aliases
- Notifications

ACCESS MANAGEMENT

- App authorizations
- App passwords**

SECURITY

- SSH keys
- Two-step verification
- Sessions
- Audit log

FEATURES

- Labs

Add app password

Details

Label* Jenkins

Permissions

Category	Read	Write	Admin
Account	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Workspace membership	<input type="checkbox"/>	<input type="checkbox"/>	
Projects	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Repositories	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Pull requests	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
Issues	<input type="checkbox"/>	<input type="checkbox"/>	
Wikis	<input type="checkbox"/>		
Snippets	<input type="checkbox"/>	<input type="checkbox"/>	
Webhooks	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
Pipelines	<input type="checkbox"/>	<input type="checkbox"/>	
Runners	<input type="checkbox"/>	<input checked="" type="checkbox"/>	

In Jenkins, create credentials for bitbucket, the ID section is used for you to reference in Jenkinsfile, not the ID in your repo account, and also create your AWS credential.

Username with password

Scope ?

Global (Jenkins, nodes, items, all child items, etc)

Username ?

JamesLSHS

Treat username as secret ?

Password ?

.....

ID ?

Bitbucket

T	P	Store ↓	Domain	ID
		System	(global)	AWS
		System	(global)	Bitbucket

To automate the pipeline, we need to add a webhook. When new codes are pushed to the repo, it will trigger the pipeline.

The screenshot shows the Jenkins 'Add new webhook' configuration page. The left sidebar has 'Webhooks' selected. The main form includes fields for 'Title' (Jenkins), 'URL' (http://ec2-13-211-206-11.ap-southeast-2.compute.amazonaws.com/bitbucket-hoos/), 'Status' (Active checked), 'SSL/TLS' (Skip certificate verification checked), and 'Triggers' (Push checked). The URL field is highlighted with a blue border.

Go to Global Tool Configuration, and set Node Js as follows.

The screenshot shows the Jenkins 'Global Tool Configuration' page for NodeJS. It includes fields for 'Name' (nodejs 16), 'Install automatically' (checked), 'Version' (NodeJS 16.10.0), and 'Global npm packages to install' (yarn).

Now, we can create our first pipeline for the front end.

The screenshot shows the Jenkins Pipeline documentation page. It defines a Pipeline as 'Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.' A circular icon with a gear and pipeline symbol is shown to the left of the text.

SCM ?

Git

Repositories ?

Repository URL ?

<https://JamesLSHS@bitbucket.org/jamesaws/fe.techscrum.git>

Credentials ?

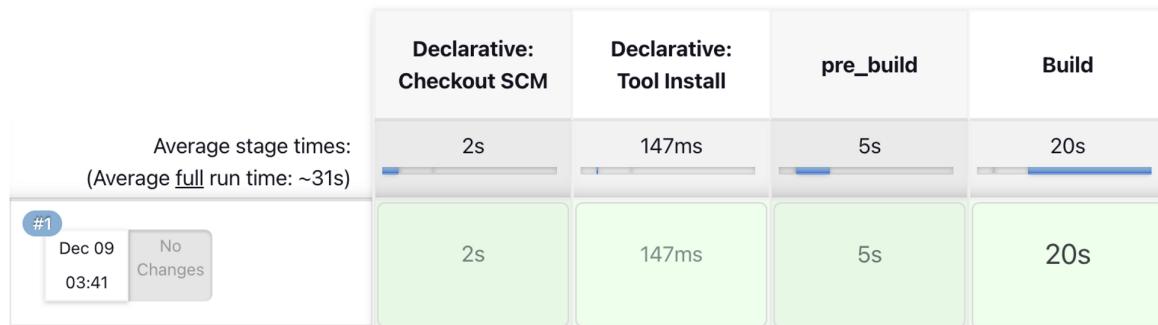
JamesLSHS /*****

+ Add

Advanced...

Pipeline Techscrum-frontend

Stage View



All artifacts have been successfully uploaded to the AWS S3 bucket.

Objects (8)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 Inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

[Actions ▾](#) [Create folder](#)

Find objects by prefix [Show versions](#) [Show preview](#) [Edit](#) [Delete](#)

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	asset-manifest.json	json	November 28, 2022, 14:39:17 (UTC+11:00)	4.3 KB	Standard
<input type="checkbox"/>	favicon.ico	ico	November 28, 2022, 14:39:16 (UTC+11:00)	4.1 KB	Standard
<input type="checkbox"/>	index.html	html	November 28, 2022, 14:39:15 (UTC+11:00)	645.0 B	Standard
<input type="checkbox"/>	logo192.png	png	November 28, 2022, 14:39:19 (UTC+11:00)	5.2 KB	Standard
<input type="checkbox"/>	logo512.png	png	November 28, 2022, 14:39:17 (UTC+11:00)	9.4 KB	Standard
<input type="checkbox"/>	manifest.json	json	November 28, 2022, 14:39:19 (UTC+11:00)	492.0 B	Standard
<input type="checkbox"/>	robots.txt	txt	November 28, 2022, 14:39:18 (UTC+11:00)	67.0 B	Standard
<input type="checkbox"/>	static/	Folder	-	-	-

Backend pipeline

For the backend, we want to check code quality with sonarqube, check the docker image vulnerability with trivy, and then push the docker image to AWS ECR, update the manifest file in Github and Argocd will update the deployment for AWS EKS.

Prerequisites

1. A SonarQube and PostgreSQL docker-compose file.



Techscrum-Jenkins-Server-Terraform.git
Jamessihang



Connect to GitHub to update

For the docker-compose file, find it in my git repo.

Create a project in SonarQube.

Techscrum ⭐ james-aws 🔒

Overview Issues Security Hotspots Measures Code Activity Project Settings ▾ Project Information

How do you want to analyze your repository?

Do you want to integrate with your favorite CI? Choose one of the following tutorials.

- With Jenkins
- With GitHub Actions
- With Bitbucket Pipelines
- With GitLab CI
- With Azure Pipelines
- Other CI

Are you just testing or have an advanced use-case? Analyze your project locally.

- Locally

Select integrate with Jenkins, and follow the step to create a pipeline.

Techscrum ⭐ james-aws 🔒

Overview Issues Security Hotspots Measures Code Activity Project Settings ▾ Project Information

How do you want to analyze your repository?

Do you want to integrate with your favorite CI? Choose one of the following tutorials.

- With Jenkins
- With GitHub Actions
- With Bitbucket Pipelines
- With GitLab CI
- With Azure Pipelines
- Other CI

Are you just testing or have an advanced use-case? Analyze your project locally.

- Locally

Generate a token for later use.

The screenshot shows the SonarQube Tokens page. At the top, there's a navigation bar with links for Projects, Issues, Rules, Quality Profiles, Quality Gates, Administration, and a search bar. Below the navigation is a user profile section with a green 'A' icon and the role 'Administrator'. To the right are links for Profile, Security, Notifications, and Projects.

The main content area is titled 'Tokens'. It contains a section for generating new tokens, where fields for Name, Type (set to 'Project'), and Expires in (set to '30 days') are shown. A button labeled 'Generate' is present. A success message box displays: 'New token "jenkins" has been created. Make sure you copy it now, you won't be able to see it again!' with a 'Copy' button and a token ID: 'sqp_d4cb80352956501695899201cc60b29634763ee1'.

A table lists existing tokens:

Name	Type	Project	Last use	Created	Expiration
jenkins	Project	Techscrum	Never	December 20, 2022	-

A red 'Revoke' button is located at the bottom right of the table.

In sonarqube set a webhook, this can enable QualityGate to send code test result back to Jenkins.

The screenshot shows the SonarQube Administration Webhooks page. The navigation bar includes links for Configuration, Security, Projects, System, and Marketplace, with 'Administration' selected. A search bar and a user profile with a green 'A' icon are also present.

The 'Webhooks' section is displayed. It says: 'Webhooks are used to notify external services when a project analysis is done. An HTTP POST request including a JSON payload is sent to each of the provided URLs. Learn more in the [Webhooks documentation](#).'. A 'Create' button is available.

A table lists the configured webhook:

Name	URL	Secret?	Last delivery
Jenkins	http://ec2-3-26-76-194.ap-southeast-2.compute.amazonaws.com:8080/sonarqube-webhook/	No	Never

2. A Jenkins Server

For backend plugins, we need to install Docker Docker Pipeline and sonarscanner on Jenkins.

On Jenkins server we need to install aws cli.

The screenshot shows the Jenkins plugin manager interface. The top navigation bar includes 'Install', 'Name ↓', and 'Released' filters. A sidebar on the left lists categories: Cloud Providers, Cluster Management, docker, and Jenkins Core.

The 'Docker' plugin is currently selected, showing version 1.2.10. It has a status badge with a checkmark and the text 'This plugin integrates Jenkins with Docker'. A note below states: 'This plugin is up for adoption! We are looking for new maintainers. Visit our [Adopt a Plugin](#) initiative for more information.' with a timestamp '3 mo 1 day ago'.

The 'Docker Pipeline' plugin is listed below, showing version 563.vd5d2e5c4007f. It has a status badge with a checkmark and the text 'Build and use Docker containers from pipelines.' A note below states: 'This plugin is up for adoption! We are looking for new maintainers. Visit our [Adopt a Plugin](#) initiative for more information.' with a timestamp '3 mo 1 day ago'.

The screenshot shows the Jenkins Marketplace page for the "SonarQube Scanner" plugin. The plugin version is 2.15, released 1 month and 5 days ago. It is categorized under "External Site/Tool Integrations" and "Build Reports". A brief description states: "This plugin allows an easy integration of SonarQube, the open source platform for Continuous Inspection of code quality."

In Jenkins, go to Configure System, add credential for SonarQube as Secret text, paste the SonarQube token in secret, and enable environment variables.

The screenshot shows the Jenkins credential configuration dialog for SonarQube. The fields are as follows:

- Secret text:** A dropdown menu.
- Scope:** Global (Jenkins, nodes, items, all child items, etc).
- Secret:** A text input field containing a masked password.
- ID:** SonarQ.
- Description:** SonarQ.

At the bottom are "Add" and "Cancel" buttons.

SonarQube servers

If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

Environment variables Enable injection of SonarQube server configuration as build environment variables

SonarQube installations

List of SonarQube installations

The screenshot shows the SonarQube server configuration dialog. It contains two sections:

- Name:** sonarqube-9.8 (with a red X icon to its right).
- Server URL:** Default is http://localhost:9000.

Go to Global Tool Configuration, set as below.

SonarQube Scanner installations

List of SonarQube Scanner installations on this system

SonarQube Scanner	
Name	<input type="text" value="SonarScanner"/>
<input checked="" type="checkbox"/> Install automatically ?	
<input type="checkbox"/> Install from Maven Central	
Version	<input type="text" value="SonarQube Scanner 4.7.0.2747"/>
<input type="button" value="Add Installer ▾"/>	

3. In AWS, attach an IAM role with ECR full access for your Jenkins EC2 instance, and configure Jenkins as a docker group user.

jenkins_ecr_role

Allows EC2 instances to call AWS services on your behalf.

Summary

Creation date	ARN	Instance profile ARN
December 28, 2022, 15:39 (UTC+11:00)	<input type="text" value="arn:aws:iam::527423341490:role/jenkins_ecr_role"/>	<input type="text" value="arn:aws:iam::527423341490:instance-profile/jenkins_ecr_role"/>
Last activity	Maximum session duration	
None	1 hour	

Permissions

Trust relationships

Tags

Access Advisor

Revoke sessions

Permissions policies (1) Info

You can attach up to 10 managed policies.

< 1 >

Policy name

Type

Description

AmazonEC2ContainerRegistryFullAccess

AWS managed

Provides administrative access t

Set up a pipeline job for backend.

S	W	Name ↓	Last Success	Last Failure	Last Duration
<input type="button" value=""/>	<input type="button" value=""/>	techscrum-backend	N/A	N/A	N/A <input type="button" value=""/>

Set up another job called update manifest, this job name should match the one in your Jenkinsfile, triggered by the backend techscrum pipeline, and update deployment file in Github. Remember to set up credential in Jenkins for Github.

Enter an item name

updatemanifest
» Required field

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a

Also, this job should be parameterized, it will pass the docker tag from the backend pipeline and override the default value.

- Discard old builds ?
- Do not allow concurrent builds
- Do not allow the pipeline to resume if the controller restarts
- GitHub project
- Pipeline speed/durability override ?
- Preserve stashes from completed builds ?
- This project is parameterized ?

String Parameter ?

Name ?
DOCKERTAG

Default Value ?
latest

Save Apply

Now, we can see the code test result in the backend pipeline and we can also access the report in SonarQube Server.

The screenshot shows the SonarQube project dashboard for the 'Techscrum' project. At the top, there's a navigation bar with links for Projects, Issues, Rules, Quality Profiles, Quality Gates, and Administration. A search bar and a user profile icon are also present. Below the header, the project name 'Techscrum' is displayed along with its GitHub icon and a branch named 'james-aws'. A message indicates the last analysis had 1 warning on December 21, 2022, at 11:15 AM. The main content area has tabs for Overview, Issues, Security Hotspots, Measures, Code, and Activity. The Overview tab is selected. On the left, a large green box indicates 'Passed' with the message 'All conditions passed.' In the center, under 'MEASURES', there are two sections: 'New Code' (since December 21, ... Started 25 minutes ago) and 'Overall Code'. Below these are sections for 'Reliability' (A) and 'Security' (A). The bottom of the dashboard features a footer with links for Project Settings and Project Information.

4. Install Argocd

```
#Reference https://argo-cd.readthedocs.io/en/stable/getting\_started/
kubectl create namespace argocd
kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml

# Install argocd cli
brew install argocd

#Make sure argocd in running in the pod
kubectl get pod -n argocd
kubectl get svc -n argocd
```

(base) ~ kubectl get pod -n argocd	READY	STATUS	RESTARTS	AGE
NAME				
argocd-application-controller-0	1/1	Running	0	4m52s
argocd-applicationset-controller-74575b6959-hhlrm	1/1	Running	0	4m52s
argocd-dex-server-64897989f8-d5fd6	1/1	Running	0	4m52s
argocd-notifications-controller-566bc99494-wwl6l	1/1	Running	0	4m52s
argocd-redis-79c755c747-vdtgp	1/1	Running	0	4m52s
argocd-repo-server-bc9c646dc-q7fcfcd	1/1	Running	0	4m52s
argocd-server-757fdbdb4d7-4cnw9	1/1	Running	0	4m52s
(base) ~ kubectl get svc -n argocd				
NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)
AGE				
argocd-applicationset-controller	ClusterIP	10.98.235.126	<none>	7000/TCP,8080/TCP
5m28s				
argocd-dex-server	ClusterIP	10.99.53.237	<none>	5556/TCP,5557/TCP,5558
/TCP 5m28s				
argocd-metrics	ClusterIP	10.97.25.248	<none>	8082/TCP
5m28s				
argocd-notifications-controller-metrics	ClusterIP	10.109.48.86	<none>	9001/TCP
5m28s				
argocd-redis	ClusterIP	10.108.82.185	<none>	6379/TCP
5m28s				
argocd-repo-server	ClusterIP	10.99.212.206	<none>	8081/TCP,8084/TCP
5m28s				
argocd-server	ClusterIP	10.99.105.246	<none>	80/TCP,443/TCP
5m28s				
argocd-server-metrics	ClusterIP	10.110.12.39	<none>	8083/TCP
5m28s				

#Get the UI with Service Type Load Balancer

```
kubectl patch svc argocd-server -n argocd -p '{"spec": {"type": "LoadBalancer"}}'
```

#If you use minikube get the url use command:

```
minikube service argocd-server -n argocd --url
```

#Or port forwarding :

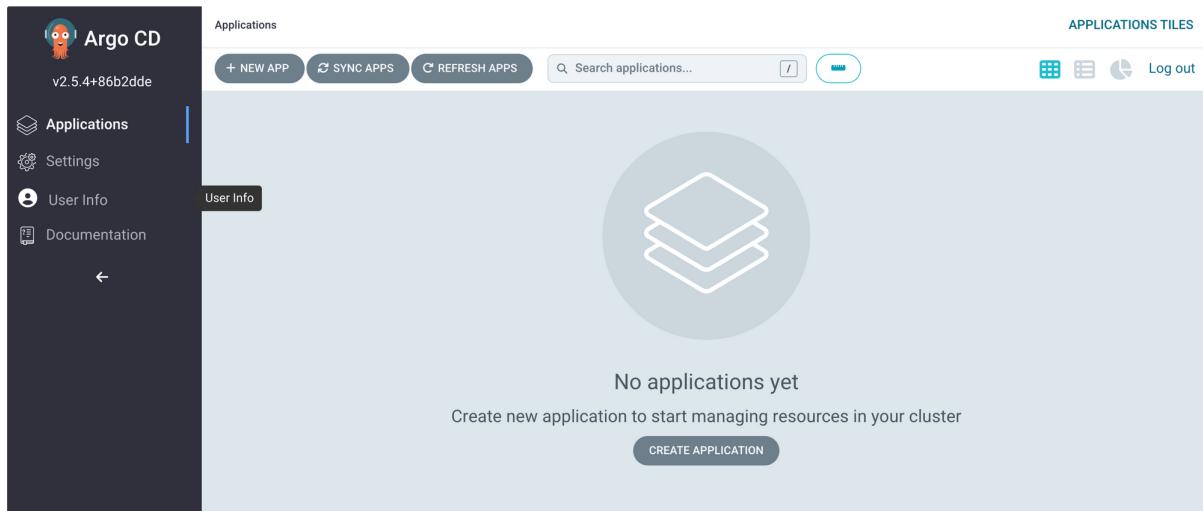
```
kubectl port-forward svc/argocd-server -n argocd 8080:443
```

#Go to localhost:8080

#Username is admin

#To get the password, run the command:

```
kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath=".data.password" | base64 -d; echo
```



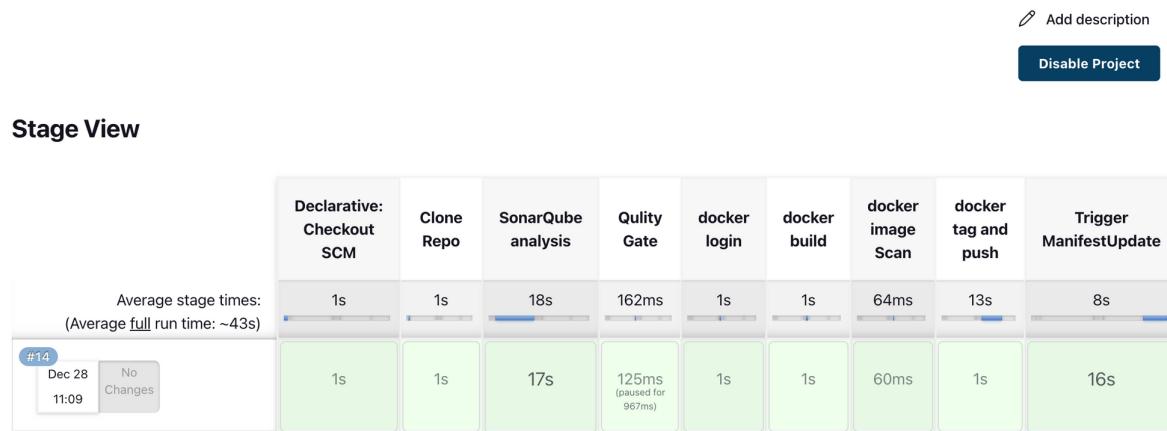
Create a new application.

Edit the Source, the URL should be where our deployment file is saved.

Edit the Destination.

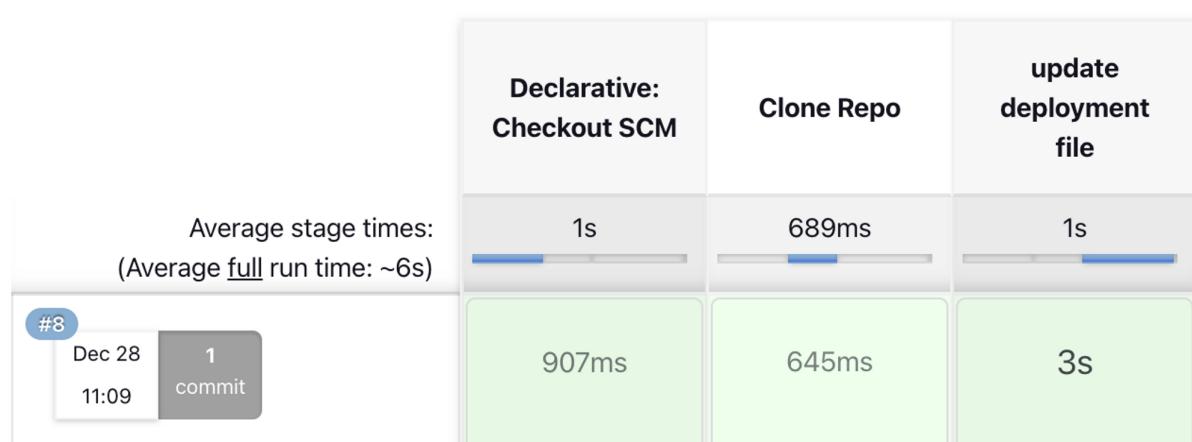
Now the backend automation is all set, whenever the new code is submitted into the code repository, Jenkins will build new docker image and push it to AWS ECR, and update deployment file in Github, at last ArgoCD will update the new deployment to our EKS cluster.

Pipeline techscrum-backend



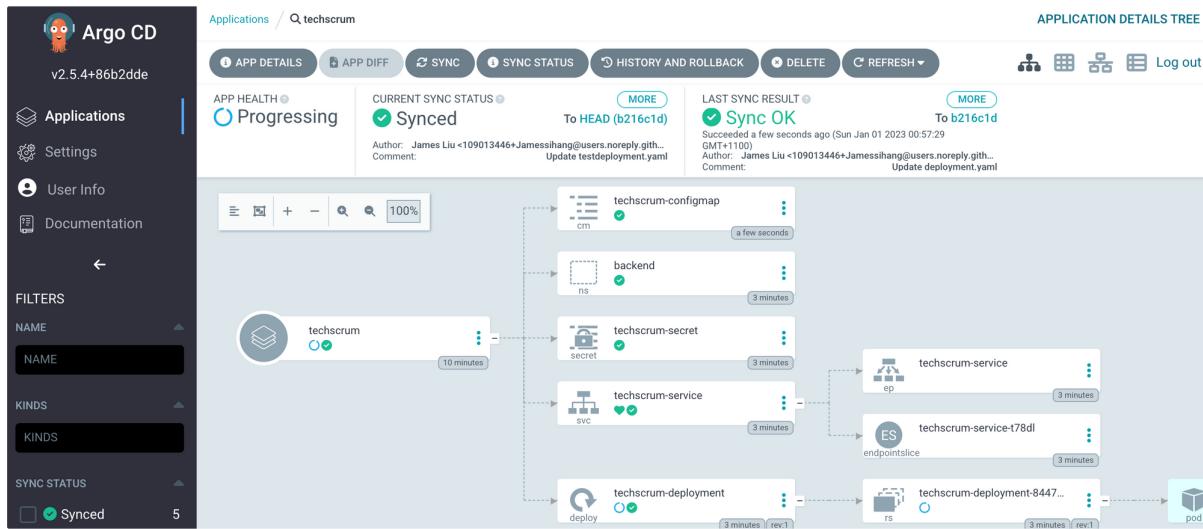
Pipeline updatemanifest

Stage View



```
app.kubernetes.io/name: techscrum
spec:
  containers:
  - name: techscrum-backend
    image: 527423341490.dkr.ecr.ap-southeast-2.amazonaws.com/techscrum:latest
    ports:
    - containerPort: 8000
      protocol: TCP
    imagePullPolicy: Always
    resources:
      limits:
        cpu: 1500m
      requests:
        cpu: 1000m
app.kubernetes.io/name: techscrum
spec:
  containers:
  - name: techscrum-backend
    image: 527423341490.dkr.ecr.ap-southeast-2.amazonaws.com/techscrum:14
    ports:
    - containerPort: 8000
      protocol: TCP
    imagePullPolicy: Always
    resources:
      limits:
        cpu: 1500m
      requests:
        cpu: 1000m
```

Images (2)							<input type="button" value="C"/>	<input type="button" value="Delete"/>	<input type="button" value="Details"/>	<input type="button" value="Scan"/>
	<input type="text"/> Search artifacts						<	1	>	↻
	Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest		Scan status	Vulnerabilities	
<input type="checkbox"/>	14	Image	December 28, 2022, 22:09:44 (UTC+11)	464.82	<input type="button" value="Copy URI"/>	<input type="button" value="sha256:8cc79f0a99b2f38..."/>	-	-	-	
<input type="checkbox"/>	latest	Image	December 20, 2022, 09:20:05 (UTC+11)	464.77	<input type="button" value="Copy URI"/>	<input type="button" value="sha256:49473269b1a288..."/>	-	-	-	



EKS

Use terraform to generate the basic infrastructures for our project, also will use helm to install tools like Grafana, ElasticSearch, Kiabana, and Fluentd. Furthermore, I will configure the AWS Application Load Balancer Controller using terraform.

Check out the code in my Github.



Techscrum-k8s.git
Jamessihang



Connect to GitHub to update

Connect to EKS cluster

Make sure you have aws cli and helm installed locally. With command:

```
aws eks --region <region> update-kubeconfig --name <cluster_name>
```

For my case:

```
aws eks --region ap-southeast-2 update-kubeconfig --name Techscrum-k8s
```

Reference Reading:

<https://aws.amazon.com/premiumsupport/knowledge-center/eks-cluster-connection/>

We want backend service can be load balanced with alb, so we need to configure the **OpenID Connect (OIDC)**, and create an IAM Role with necessary policy, which gives it **IRSA (IAM Role for Service Account)**.

```
..} AWSLoadBalancerController.json > ...
1  {
2    "Version": "2012-10-17",
3    "Statement": [
4      {
5        "Effect": "Allow",
6        "Action": [
7          "iam:CreateServiceLinkedRole"
8        ],
9        "Resource": "*",
10       "Condition": {
11         "StringEquals": {
12           "iam:AWSServiceName": "elasticloadbalancing.
13             amazonaws.com"
14         }
15       },
16       {
17         "Effect": "Allow",
18         "Action": [
19           "ec2:DescribeAccountAttributes",
20           "ec2:DescribeAddresses",
21           "ec2:DescribeAvailabilityZones",
22           "ec2:DescribeInternetGateways",
23           "ec2:DescribeVpcs",
24           "ec2:DescribeVpcPeeringConnections",
25           "ec2:DescribeSubnets",
26           "ec2:DescribeSecurityGroups",
27           "ec2:DescribeInstances",
28           "ec2:DescribeNetworkInterfaces",
29           "ec2:DescribeTags",
30           "ec2:GetCoipPoolUsage",
31           "ec2:DescribeCoipPools",
32           "elasticloadbalancing:DescribeLoadBalancers",
33           "elasticloadbalancing:DescribeLoadBalancerAttributes"
34         }
35       }
36     ]
37   }
38 }
```

AWS Application Load Balancer Controller has created a alb for backend service with backend application deployment.

```
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: alb-ingress
  annotations:
    kubernetes.io/ingress.class: alb
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/target-type: instance
    alb.ingress.kubernetes.io/certificate-arn:
      "arn:aws:acm:ap-southeast-2:527423341490:certificate/
      7527d578-50e5-4d4e-9005-4cadea5d9e7e"
    alb.ingress.kubernetes.io/listen-ports: '[{"HTTP": 80},
      {"HTTP": 8000}, {"HTTPS":443}]'
    alb.ingress.kubernetes.io/ssl-redirect: '443'
    alb.ingress.kubernetes.io/healthcheck-port: /api/v1/users
    alb.ingress.kubernetes.io/success-codes: "200-399"
spec:
  rules:
    - host: api.jamestechscrum.com
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: techscrum
                port:
                  number: 8000
```

EC2 > Load balancers > k8s-default-albingre-9f860811d3

k8s-default-albingre-9f860811d3															
Actions ▾															
▼ Details <p>arn:aws:elasticloadbalancing:ap-southeast-2:527423341490:loadbalancer/app/k8s-default-albingre-9f860811d3/cc69a47c05b7198d</p> <table border="1"> <tr> <td>Load balancer type Application</td> <td>DNS name k8s-default-albingre-9f860811d3-1948825001.ap-southeast-2.elb.amazonaws.com (A Record)</td> <td>Status Provisioning</td> <td>VPC vpc-035060eb8f746f60b</td> </tr> <tr> <td>IP address type IPv4</td> <td>Scheme Internet-facing</td> <td>Availability Zones subnet-098af9247d4bc8d8b ap-southeast-2b (apse2-az1) subnet-056377b271c2cfac8 ap-southeast-2a (apse2-az3) </td> <td>Hosted Zone Z1GM3OXH4ZPM65</td> </tr> <tr> <td>Created At December 29, 2022, 18:01 (UTC+11:00)</td> <td colspan="3"></td> </tr> </table>				Load balancer type Application	DNS name k8s-default-albingre-9f860811d3-1948825001.ap-southeast-2.elb.amazonaws.com (A Record)	Status Provisioning	VPC vpc-035060eb8f746f60b	IP address type IPv4	Scheme Internet-facing	Availability Zones subnet-098af9247d4bc8d8b ap-southeast-2b (apse2-az1) subnet-056377b271c2cfac8 ap-southeast-2a (apse2-az3)	Hosted Zone Z1GM3OXH4ZPM65	Created At December 29, 2022, 18:01 (UTC+11:00)			
Load balancer type Application	DNS name k8s-default-albingre-9f860811d3-1948825001.ap-southeast-2.elb.amazonaws.com (A Record)	Status Provisioning	VPC vpc-035060eb8f746f60b												
IP address type IPv4	Scheme Internet-facing	Availability Zones subnet-098af9247d4bc8d8b ap-southeast-2b (apse2-az1) subnet-056377b271c2cfac8 ap-southeast-2a (apse2-az3)	Hosted Zone Z1GM3OXH4ZPM65												
Created At December 29, 2022, 18:01 (UTC+11:00)															

Monitoring:

1. Install Prometheus



Techscrum-k8s.git

Jamessihang



Connect to GitHub to update

`kubectl create ns monitoring`

`cd into prometheus folder`

`kubectl apply -f ./`

```
(base) ~/Desktop/Techscrum-k8s/K8S/prometheus (main ✓) kubectl apply -f ./clusterrole.rbac.authorization.k8s.io/prometheus created
clusterrolebinding.rbac.authorization.k8s.io/prometheus created
configmap/prometheus-server-conf created
deployment.apps/prometheus-deployment created
service/prometheus-service created
```

`cd into kube-state-metrics-configs folder`

`kubectl apply -f ./`

```
(base) ~/Desktop/Techscrum-k8s/k8s/_kube-state-metrics-configs (main ✘) ls
cluster-role-binding.yaml cluster-role.yaml deployment.yaml service-account.yaml service.yaml
(base) ~/Desktop/Techscrum-k8s/k8s/_kube-state-metrics-configs (main ✘) kubectl apply -f ./clusterrolebinding.rbac.authorization.k8s.io/kube-state-metrics created
clusterrole.rbac.authorization.k8s.io/kube-state-metrics created
deployment.apps/kube-state-metrics created
serviceaccount/kube-state-metrics created
service/kube-state-metrics created
(base) ~/Desktop/Techscrum-k8s/k8s/_kube-state-metrics-configs (main ✘)
```

Prometheus is running as a nodeport service, access it with **nodeIP:nodePort**, for example: <http://192.168.49.2:30001>

Or with minikube, you can expose service with command:

`minikube service prometheus-service -n monitoring`

The screenshot shows the Prometheus Targets page with two sections: **kube-state-metrics (1/1 up)** and **kubernetes-apiservers (1/1 up)**.

kube-state-metrics (1/1 up)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://kube-state-metrics.kube-system.svc.cluster.local:8080/metrics	UP	Instances="kube-state-metrics.kube-system.svc.cluster.local:8080" job="kube-state-metrics"	599.000ms ago	6.690ms	

kubernetes-apiservers (1/1 up)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
https://192.168.49.2:8443/metrics	UP	instance="192.168.49.2:8443" job="kubernetes-apiservers"	1.62s ago	96.934ms	

2. Install Grafana

For monitor visualization, I will use helm to install Grafana, with command:

```
helm repo add grafana https://grafana.github.io/helm-charts
```

```
helm repo update
```

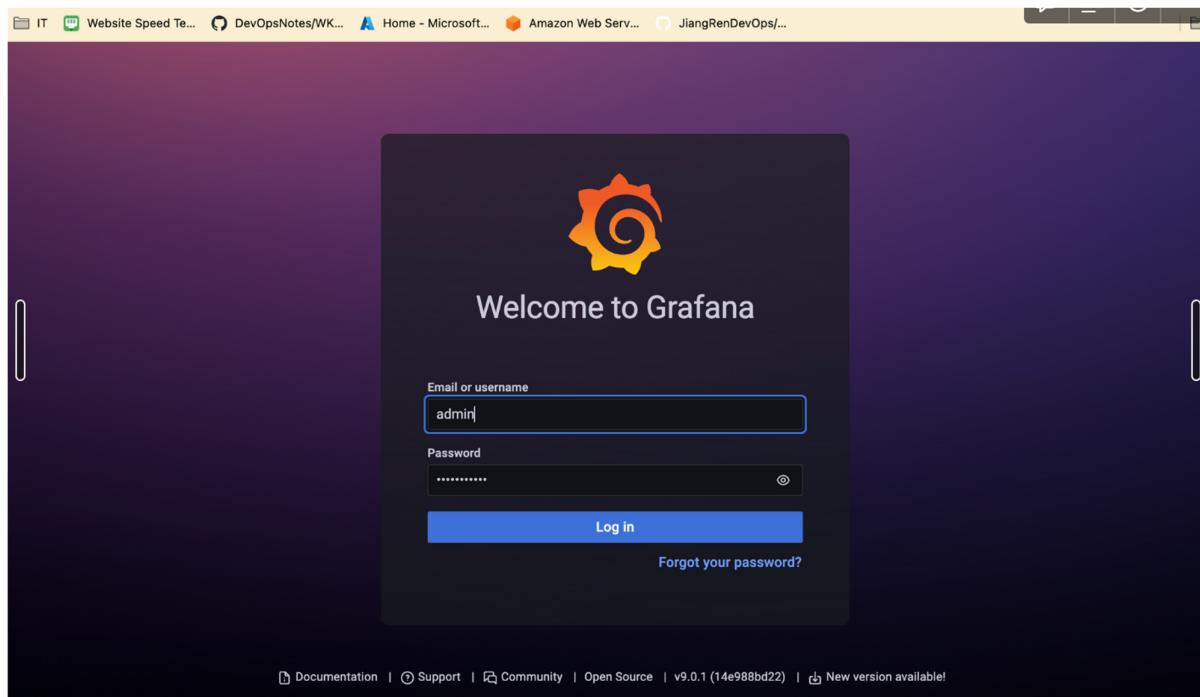
```
helm install grafana grafana/grafana -n monitoring
```

Expose the service with Nodeport:

```
kubectl expose service grafana -n monitoring --type=NodePort --target-port=3000 --name=grafana-ext
```

If you use minikube, find external url with command:

```
minikube service grafana-ext -n monitoring
```



Default username is admin, get the password with command:

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

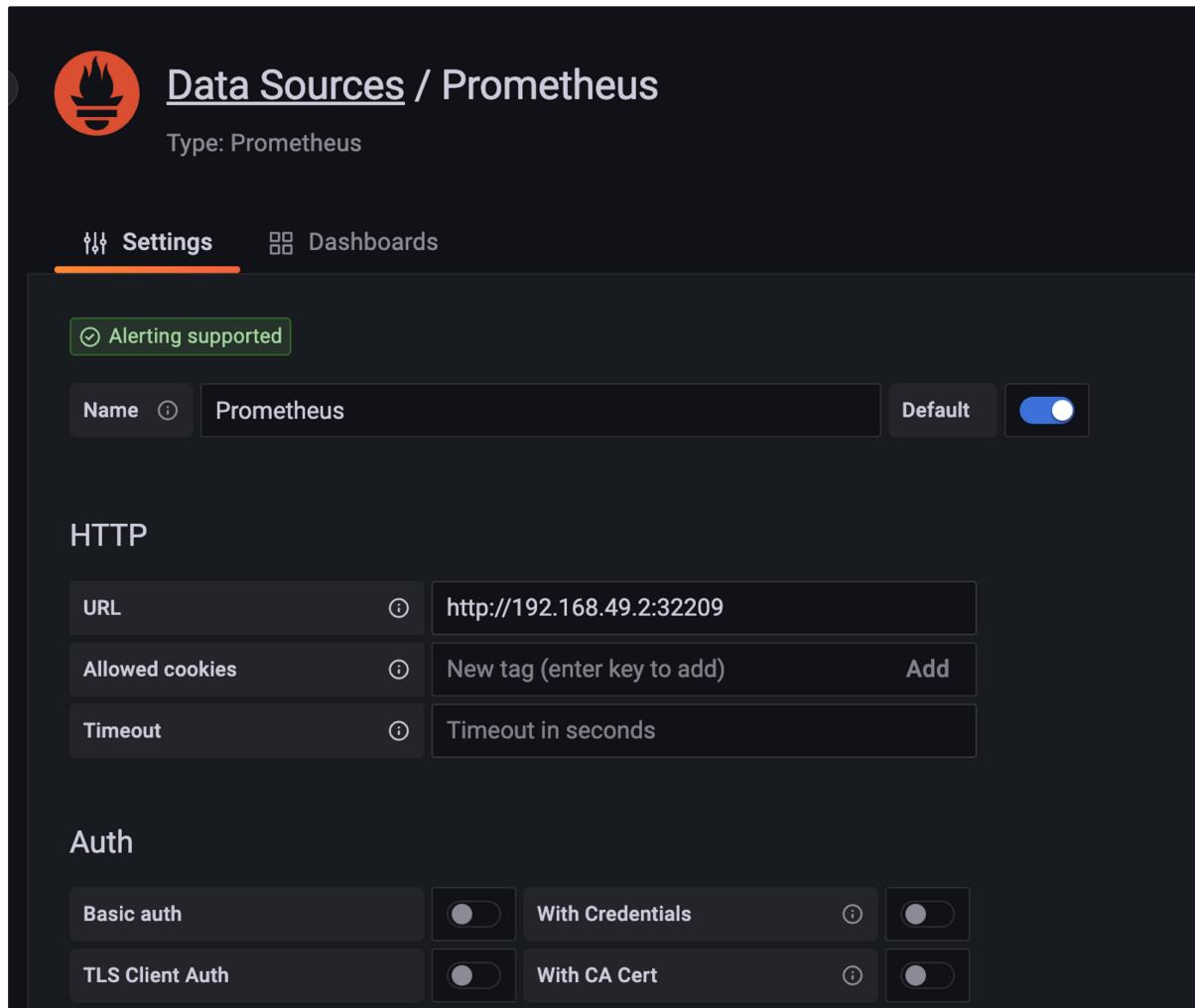
Or you want to get full details, you can run command:

```
kubectl get secret --namespace monitoring grafana -o yaml
```

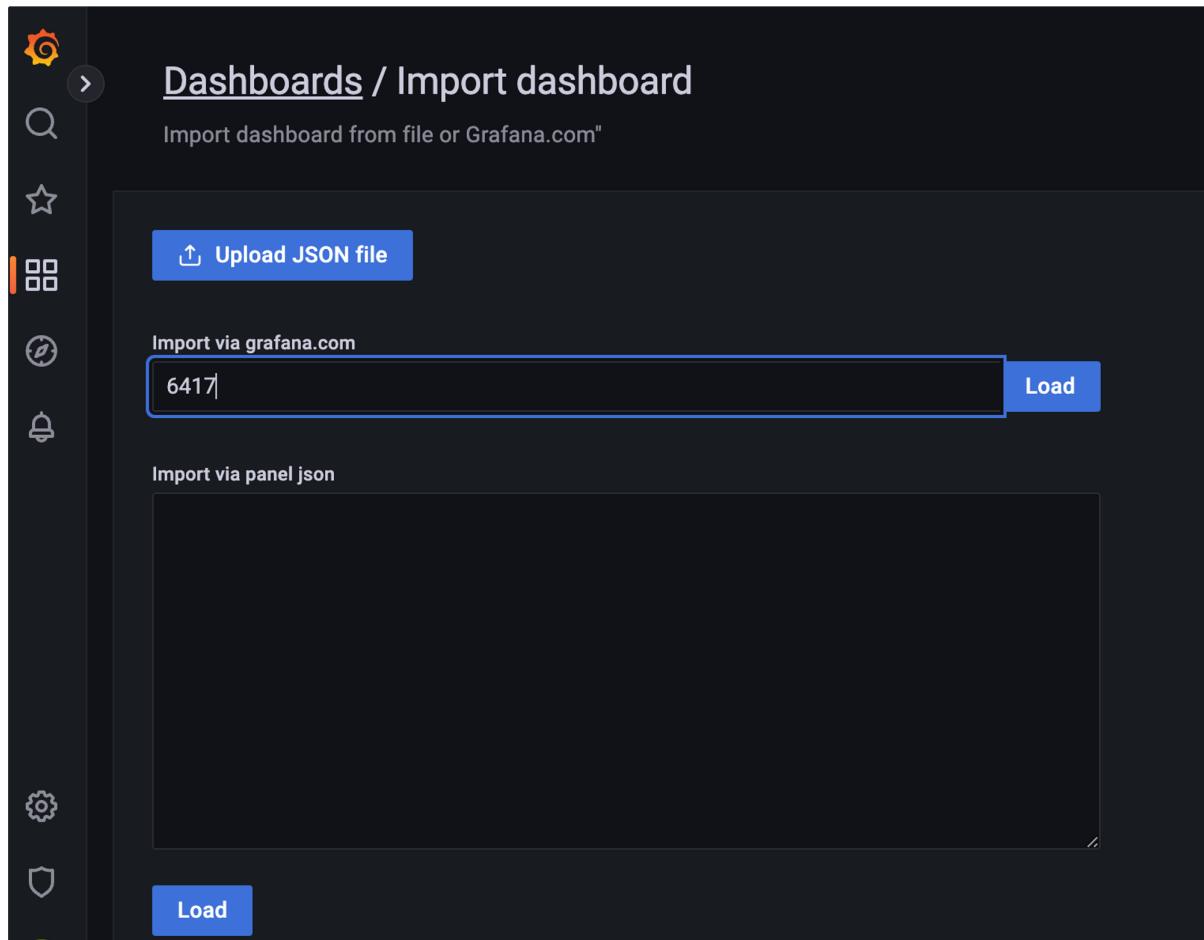
The username and password is base64 encoded.

```
jamesliu@kous-MacBook-Pro:~ % Last login: Thu Dec 29 23:33:42 on ttys001
(base) ~ kubectl get secret --namespace monitoring grafana -o yaml
apiVersion: v1
data:
  admin-password: SWRYZzNiajQzS0JvSmhsRWdIbnpsUEx50XU5TThsd2drZDNPSEFDRQ==
  admin-user: YWRtaW4=
  ldap-toml: ""
kind: Secret
metadata:
  annotations:
    meta.helm.sh/release-name: grafana
    meta.helm.sh/release-namespace: monitoring
  creationTimestamp: "2022-12-29T12:45:13Z"
  labels:
    app.kubernetes.io/instance: grafana
    app.kubernetes.io/managed-by: Helm
    app.kubernetes.io/name: grafana
    app.kubernetes.io/version: 9.3.1
    helm.sh/chart: grafana-6.48.2
  name: grafana
  namespace: monitoring
  resourceVersion: "119248"
  uid: 7ace21db-932d-40ea-aff3-1f09f0321115
type: Opaque
(base) ~ %
```

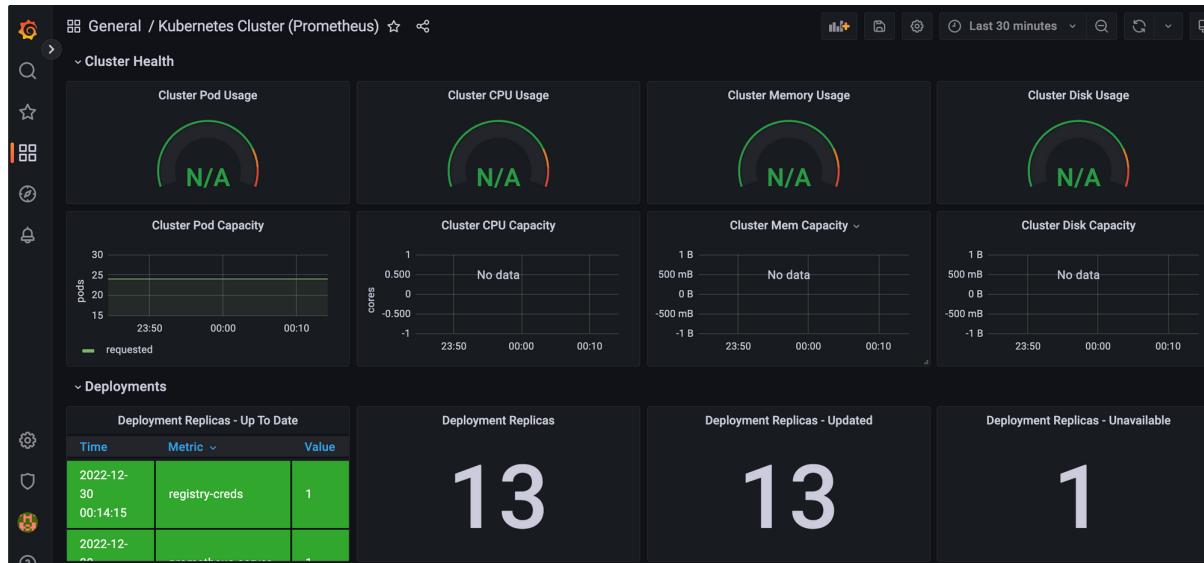
In Grafana, add data source Prometheus, and URL for Prometheus server, and import a dash board 6417.

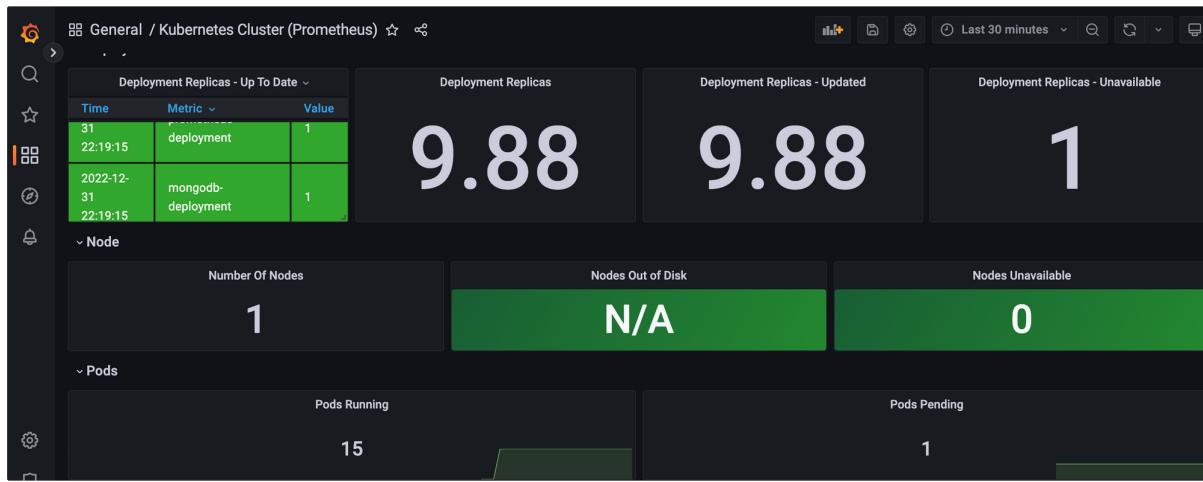


The screenshot shows the 'Data Sources / Prometheus' configuration page in Grafana. The top navigation bar includes a Prometheus icon, the title 'Data Sources / Prometheus', and the text 'Type: Prometheus'. Below the title, there are two tabs: 'Settings' (which is selected) and 'Dashboards'. A green button labeled '(?) Alerting supported' is visible. The main configuration area starts with an 'HTTP' section containing fields for 'URL' (set to 'http://192.168.49.2:32209'), 'Allowed cookies' (with a placeholder 'New tag (enter key to add)' and an 'Add' button), and 'Timeout' (set to 'Timeout in seconds'). Below this is an 'Auth' section with four toggle switches: 'Basic auth' (off), 'With Credentials' (on), 'TLS Client Auth' (off), and 'With CA Cert' (on). The entire interface has a dark theme.



Now the dashboard is all set.





3. Install Alert Manager

cd into **alert** folder

kubectl apply -f ./

```
(base) ~/Desktop/Techscrum-k8s/k8s/alert (main ✘) kubectl create -f AlertManagerConfigmap.yaml
configmap/alertmanager-config created
(base) ~/Desktop/Techscrum-k8s/alert (main ✘) kubectl create -f AlertTemplateConfigMap.yaml
configmap/alertmanager-templates created
(base) ~/Desktop/Techscrum-k8s/alert (main ✘) kubectl create -f Deployment.yaml
deployment.apps/alertmanager created
(base) ~/Desktop/Techscrum-k8s/k8s/alert (main ✘) kubectl create -f Service.yaml
service/alertmanager created
```

Check the pod and service is running.

```
(base) ~/Desktop kubectl get pod -n monitoring
NAME                               READY   STATUS    RESTARTS   AGE
alertmanager-6ffb68c68d-4jc7l      1/1     Running   0          27m
grafana-869cdb4759-7lmqd          1/1     Running   0          24m
prometheus-deployment-6df86f75b5-zztz2 1/1     Running   0          26m
(base) ~/Desktop kubectl get svc -n monitoring
NAME        TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
alertmanager   NodePort   10.103.37.245 <none>       9093:31000/TCP   27m
grafana       ClusterIP  10.101.14.96  <none>       80/TCP      25m
grafana-ext    NodePort   10.105.45.29  <none>       80:31298/TCP   25m
prometheus-service  NodePort   10.105.124.98 <none>       9090:30001/TCP   26m
(base) ~/Desktop
```

Access Alertmanager UI with **nodeIP:nodePort**, for example:

<http://192.168.49.2:31000>

Or with minikube, you can expose service with command:

minikube service alertmanager -n monitoring

Alertmanager Alerts Silences Status Settings Help

Receiver: All Silenced Inhibited

Custom matcher, e.g. `env="production"`

+ Silence

No alert groups found

Check the status for Alertmanager.

Alertmanager Alerts Silences **Status** Settings Help

Status

Uptime: 2022-12-30T23:21:33.640Z

Cluster Status

Name: 01GNJNEEP4R30R8E7BSJTQ7CYE

Status: ready

Peers:

- Name: 01GNJNEEP4R30R8E7BSJTQ7CYE
- Address: 172.17.0.5:9094

Version Information

Branch: HEAD

BuildDate: 20221222-14:48:36

BuildUser: root@521a2d62cff8

GoVersion: go1.19.4

Revision: 258fab7cdd551f2cf251ed0348f0ad7289aee789

Version: 0.25.0

In Slack we need to add a channel to receive alert from Alertmanager.

And then go to Apps, search webhook, choose incoming webhook.

Techscrum ▾

⋮ Apps

Browse Slack

Channels

general

random

techscrum

+ Add channels

Direct messages

Eden Relax Music you

jameshang

+ Add teammates

⚡ Connect your tools

With each app that you add, Slack gets even more useful. It syncs with your calendar, displays files from the cloud and brings all your work tools together.

Next, you could...

Learn more about apps Install Google Calendar Install Google Drive

Q webhook

Available apps

Incoming WebHooks Postmark Bot Cronhooks

Send data into Slack in real-time. Receive webhook notifications, status notifications, and more. Schedule on time or recurring webhooks

Integrate webhook with our channel.

Post to channel

Start by choosing a channel your incoming webhook will post messages to.

▼

or [create a new channel](#)

Add Incoming webhooks integration

By creating an incoming webhook, you agree to the [Slack API terms of service](#).

In prometheus, we have configured some rule for alerting.

```
k8s > prometheus > config-map.yaml > {} data > prometheus.rules
      io.k8s.api.core.v1.ConfigMap (v1@configmap.json)
1  apiVersion: v1
2  kind: ConfigMap
3  metadata:
4    name: prometheus-server-conf
5    labels:
6      name: prometheus-server-conf
7    namespace: monitoring
8  data:
9    prometheus.rules: |-
10      groups:
11        - name: techscrum demo alert
12          rules:
13            - alert: High Pod Memory
14              expr: sum(container_memory_usage_bytes) > 1
15              for: 1m
16              labels:
17                severity: slack
18              annotations:
19                summary: High Memory Usage
20
21            - alert: InstanceDown
22              expr: up == 0
23              for: 1m
24              annotations:
25                title: 'Instance {{ $labels.instance }} down'
26                description: '{{ $labels.instance }} of job {{ $labels.job }} has been down for more than 1 minute.'
27              labels:
28                severity: 'slack'
29
30            - alert: HostOutOfMemory
31              expr: node_memory_MemAvailable / node_memory_MemTotal * 100
32              < 25
33              for: 5m
34              annotations:
35                title: 'Host out of memory (instance {{ $labels.
```

You can also see them in prometheus UI.

Alert Details	Status	Age	Duration
alert: High Pod Memory expr: sum(container_memory_usage_bytes) > 1 for: 1m labels: severity: slack annotations: summary: High Memory Usage	OK	883.000ms ago	2.129ms
alert: InstanceDown expr: up == 0 for: 1m labels: severity: slack annotations: description: {{ \$labels.instance }} of job {{ \$labels.job }} has been down for more than 1 minute. title: Instance {{ \$labels.instance }} down	OK	881.000ms ago	2.954ms
alert: HostOutOfMemory expr: node_memory_MemAvailable / node_memory_MemTotal * 100 < 25 for: 5m labels: severity: slack annotations: description: Node memory is filling up (< 25% left)\n VALUE = {{ \$value }}\n LABELS: {{ \$labels }} title: Host out of memory (instance {{ \$labels.instance }})	OK	878.000ms ago	0.545ms

Now go to Alerts in prometheus, we can see some alerts are active.

Alerts (4 active)

- > High Pod Memory (0 active)
- > InstanceDown (2 active)
- > HostOutOfMemory (0 active)
- > HostHighCpuLoad (0 active)
- > HostOutOfDiskSpace (0 active)

Go to Alertmanager, we can also see these alerts.

Alertmanager

2 alerts

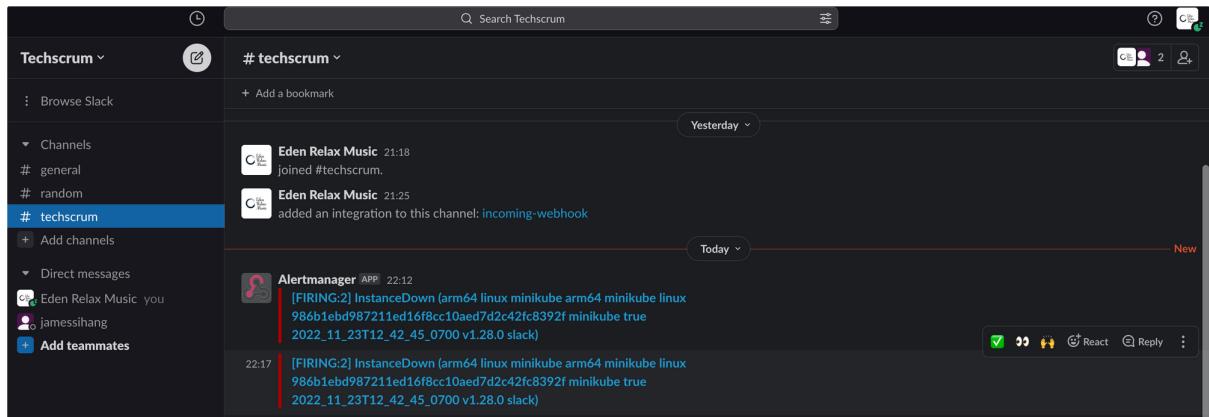
2022-12-31T11:12:12.056Z + Info ↗ Source ✖ Silence % Link

beta_kubernetes_io_arch="arm64" + beta_kubernetes_io_os="linux" + instance="minikube" + job="kubernetes-cadvisor" +
kubernetes_io_arch="arm64" + kubernetes_io_hostname="minikube" + kubernetes_io_os="linux" +
minikube_k8s_io_commit="986b1ebd987211ed16f8cc10aed7d2c42fc8392f" + minikube_k8s_io_name="minikube" + minikube_k8s_io_primary="true" +
minikube_k8s_io_updated_at="2022_11_23T12_42_45_0700" + minikube_k8s_io_version="v1.28.0" + severity="slack" +

2022-12-31T11:12:12.056Z + Info ↗ Source ✖ Silence % Link

beta_kubernetes_io_arch="arm64" + beta_kubernetes_io_os="linux" + instance="minikube" + job="kubernetes-nodes" +

In slack, we have already received those alert notifications.



Database

Install monogoDB and mongoExpress



Techscrum-k8s.git
Jamessihang



Connect to GitHub to update

kubectl create ns mongodb

cd into mongodb folder

kubectl apply -f ./

```
(base) ~/Desktop/Techscrum-k8s/k8s (main ✘) cd mongodb
(base) ~/Desktop/Techscrum-k8s/k8s/mongodb (main ✘) ls
mongodb-config.yaml      mongodb-deployment.yaml      mongoDB-secret.yaml      mongoDB-service.yaml      mongoexpress-deployment.yaml      mongoexpress-service.yaml
(base) ~/Desktop/Techscrum-k8s/k8s/mongodb (main ✘) kubectl apply -f ./configmap/mongodb-configmap created
deployment.apps/mongodb-deployment created
secret/mongoDB-secret created
service/mongoDB-service created
deployment.apps/mongo-express created
service/mongo-express-service created
```

Check the pods and services are running.

```
(base) ~/Desktop/Techscrum-k8s/k8s/mongodb (main ✘) kubectl get pod -n mongodb
NAME                               READY   STATUS    RESTARTS   AGE
mongo-express-5bf4b56f47-pgnsq     1/1    Running   1 (115s ago)  2m4s
mongodb-deployment-844789cd64-n9vhm 1/1    Running   0          2m4s
(base) ~/Desktop/Techscrum-k8s/k8s/mongodb (main ✘) kubectl get svc -n mongodb
NAME           TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
mongo-express-service   LoadBalancer  10.108.145.80  <pending>      8081:30000/TCP  2m20s
mongodb-service   ClusterIP  10.101.79.27  <none>        27017/TCP    2m20s
(base) ~/Desktop/Techscrum-k8s/k8s/mongodb (main ✘)
```

Access mongo express

I have configured Mongo express service type as loadbalancer, you can access it with external ip or if you use minikube run command:

minikube service mongo-express-service -n mongodb

Mongo Express Database

Databases		Database Name	+ Create Database
	admin		Del
	config		Del
	local		Del

Server Status

Turn on admin in config.js to view server stats!

Create a new database for Techscrum.

Mongo Express Database

Databases		Database Name	+ Create Database
	Techscrum		Del
	admin		Del
	config		Del
	local		Del

Server Status

Turn on admin in config.js to view server stats!

Type '/' for commands

Mongo Express Database: Techscrum

Collections		Collection Name	+ Create collection
	delete_me		Del

Mongo Express Database: Techscrum > Collection: db1

Viewing Collection: db1

New Document New Index

Simple Advanced

name	James Liu	String	Find
------	-----------	--------	------

No documents found.

Rename Collection

Techscrum . db1 Rename

