

# ON-PREMISE APPLICATION DEPLOYMENT

## PROJECT REPORT

Ramprasath A – ramprasathasokan@gmail.com

This project aims at solving on-premise application deployments for the enterprise clients of a leading tech corp. The requirements are managing multiple microservices running across different applications and how to deploy and manage microservices individually. As an example we are going to deploy a containerized web application namely [Parse Server Example](#).

In order to further increase the ease of operations we are going to take into account the following challenges faced in deployments too:

1. Ease of clustered enterprise level deployments
2. Incremental remotely triggered application updates
3. Easy remote debugging
4. Health Alerts and Monitoring
5. Application Security (with source code protection)
6. Disaster management

Before diving into the installation instructions, let us see what are the various tools used and what requirements does it satisfies.

### Tools Used

1. Google Cloud Platform – for infrastructure requirements
2. Docker – for setting up Rancher and Harness delegate
3. Rancher – for managing Kubernetes cluster
4. GKE (Google Kubernetes Engine) – for provisioning Kubernetes cluster
5. Kubernetes – for deploying our parser server application
6. Prometheus and Grafana – for monitoring
7. ELK stack (Elasticsearch, Logstash, Kibana) – for log management
8. Slack – for notification of alerts
9. Harness – for CD pipeline
10. Helm – for providing helm charts for parse server application
11. Velero – for backing up and restoring data in the event of a disaster

### Assumptions

I assumed it to be an actual production environment but in the interest of time, I used docker containers for running Rancher and Harness delegate. Also, I used a managed ELK stack by Bitnami Launchpad which may or may not go along with the privacy guidelines of the organization. But as far as Harness is considered, there is an option to opt for a self-hosted version.

So, essentially, I chose tools and services which can be easily run on premise so that we can ensure more security and privacy of data. Also, I assumed GCP as on-premise infrastructure.

## **Solving the challenges faced in deployments using the above tools**

### **1. Ease of clustered enterprise level deployments**

We are using Rancher which provides a rich GUI to provision, monitor, alert, notify, log a Kubernetes cluster. Also, we can rollback deployments and backup etcd data by taking snapshots.

### **2. Incremental remotely triggered application updates**

In Harness, under workflow creation we can choose rolling update strategy i.e. incremental updates strategy and we can trigger the updates remotely i.e. instead of directly interacting with the cluster through Harness.

### **3. Ease remote debugging**

We are using the ELK Stack i.e. Elasticsearch, Logstash and Kibana Stack for log management. Using the ELK stack we can retrieve a wide range of logs along with their timestamp to easily debug errors (if any) remotely. Also, updates regarding health of a node will be notified via Slack channel from Rancher which will further ease remote debugging. Also, we can use the insights from Prometheus and Grafana

### **4. Health Alerts and Monitoring**

We are using Prometheus and Grafana for monitoring our Kubernetes cluster and Rancher has a set of Health Alert groups which can be configured with Slack for notification if required.

### **5. Application Security**

Rancher and Harness both support enforcing of RBAC (Rule Based Access Control) and we are leveraging it to ensure security of our management and deployment. If we happen to use NGINX-Ingress as our load balancer we can enable OWASP ModSecurity WAF rules just by adding an annotation: `nginx.ingress.kubernetes.io/enable-modsecurity: true`. We can use code obfuscation tools to obfuscate our source code so it will be harder to reverse engineer to obtain the source code.

### **6. Disaster Management**

Disaster management deals with unexpected failure. A standby Kubernetes cluster and an up-to-date standby database with the same version of the application deployed as in the currently running cluster will help in ensuring high availability and thereby reducing the downtime in the event of a disaster.

In the case of backup, the first and foremost priority for backing up should be given to the database of the MongoDB used by our parse server. Then, we should take backups of etcd. We should backup these data regularly once every hour or once a day based on client

requirements. Most importantly we should ensure to store the backups in a highly available storage like AWS S3. In order to achieve this, we can use Velero.

After a disaster, we can recover from a disaster by restoring the latest backup of our database and application state.

## Installation / Usage Instructions

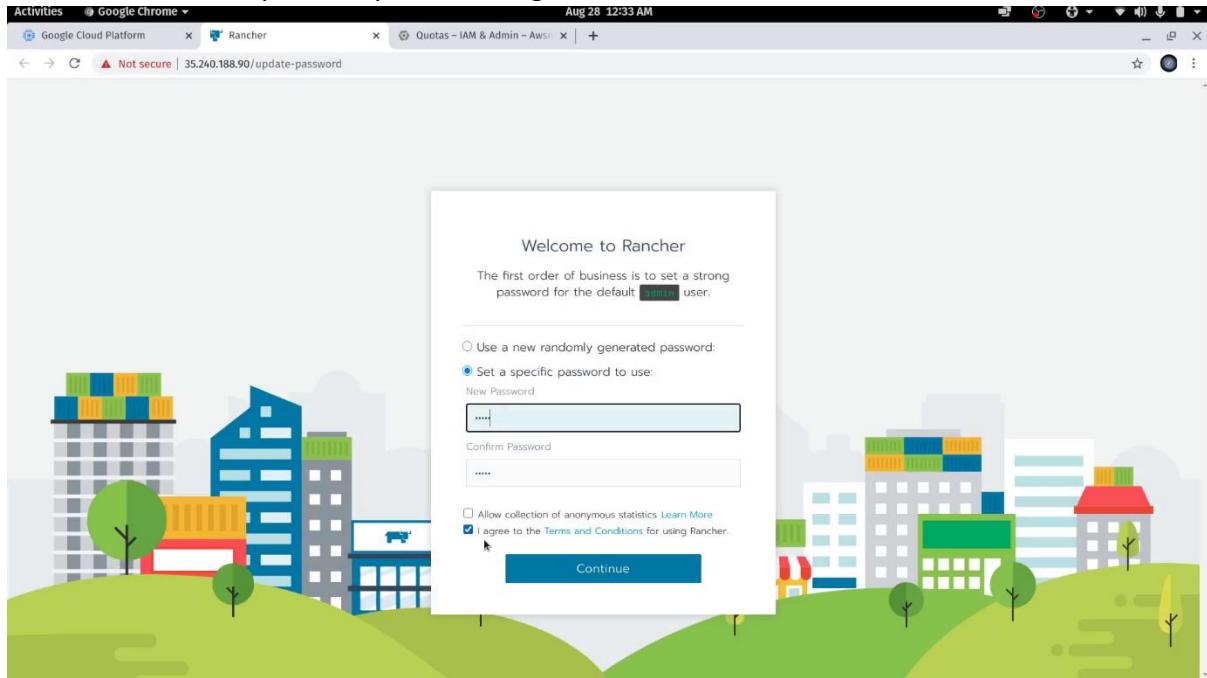
YouTube link: <https://youtu.be/FMxPRegLg-A>

GitHub link: <https://github.com/ramprasathasokan/On-Premise-Application-Deployment>

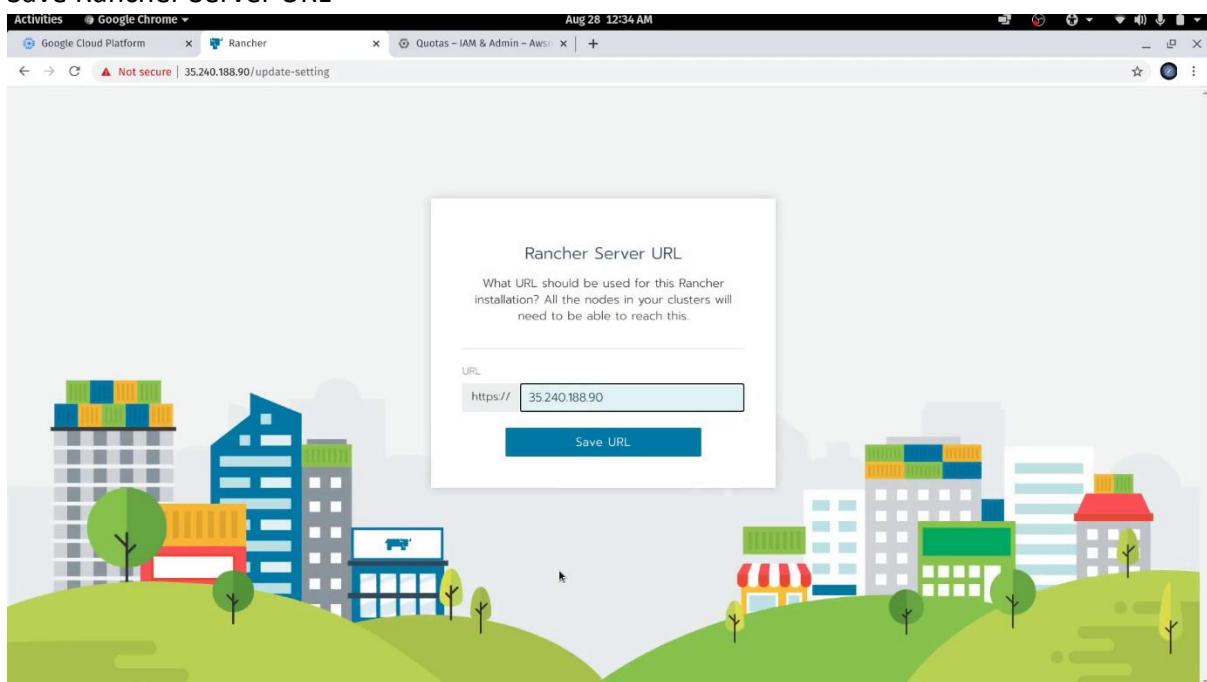
Note: Before setting up VM instances we need to create two different projects in Google Cloud Platform because each region has a quota for vCPU about 8 vCPUs. In one project we will setup Rancher, ELK Stack and Harness delegate and we will use the other project solely for Kubernetes cluster.

1. We need to setup our infrastructure, in this case, VMs. We are going to create 2 VM instance initially, one for Rancher and another one for Harness delegate. We will see about Harness delegate while setting up Harness CD pipeline. The following are specifications of each VM: n1-standard, 1vCPU, 3.75GB RAM, 50 GB Storage, Ubuntu 18.04 LTS. Select a desirable region and zone as per requirements. Enable both allow HTTP/HTTPS traffic.
2. Before clicking create instance we need to add our SSH credentials under Security tab of the instance creation page.
  - To generate SSH key: `ssh-keygen -t ecdsa -C username`
  - To copy securely: `xclip -sel clip < ~/.ssh/id_ecdsa.pub`
3. Now create instance and make sure to create another instance. Tip: we can use templates to create VMs.
4. SSH into both VMs using: `ssh -i ~/.ssh/id_ecdsa.pub username@external-ip`
5. We need to run: `sudo apt update && sudo apt upgrade -y` in both VMs and then perform a reboot: `sudo reboot`.
6. Then we need to install Docker-CE (Docker Engine) and verify installation in both VMs by running the following commands one by one:
  - `sudo apt update`
  - `sudo apt install apt-transport-https ca-certificates curl software-properties-common`
  - `curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -`
  - `sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable"`
  - `sudo apt update`
  - `apt-cache policy docker-ce`
  - `sudo apt install docker-ce`
  - `sudo systemctl status docker`
  - `sudo usermod -aG docker ${USER}`

- `sudo su - ${USER}`
  - `id -nG`
  - `docker info`
7. Now we need to install Rancher in its respective VM by running the following command:
- `sudo docker run -d --restart=unless-stopped -p 80:80 -p 443:443 -v ~/rancher-data:/var/lib/rancher rancher/rancher`
  - Note: We are mounting volume of Rancher to make the data persistent while performing reboot.
8. Then, we can access rancher using the external-ip of the instance where Rancher is installed. Then setup admin password, agree to T&C and continue.



#### 9. Save Rancher Server URL



10. Now before we go ahead and create a cluster, let's see how to add a user and implement RBAC (Role Based Access Control) with Rancher.

- Go to **Security > User > Add User**

The screenshot shows the Rancher web interface under the 'Security' tab. The 'Users' section displays a table with one row. The columns are 'State' (Active), 'Name' (Default Admin ( admin )), 'ID' (user-tctzr), and 'Local Username' (admin). There are buttons for 'Deactivate' and 'Delete' at the top of the table, and a 'Search' input field. The URL in the browser is https://35.240.188.90/g/security/accounts/users.

- Then enter username, password, display name and set global permissions for enforcing RBAC.

The screenshot shows the 'Add User' form. It has fields for 'Username' (std-user-1), 'Password' (\*\*\*\*\*), 'Display Name' (std-user-1), and 'Global Permissions'. The 'Global Permissions' section contains three checkboxes: 'Administrator' (unchecked), 'Standard User' (checked), and 'User-Base' (unchecked). The URL in the browser is https://35.240.188.90/g/security/accounts/add.

- Apart from the roles defined by Rancher, we can also add custom roles.

The screenshot shows the 'Quotas - IAM & Admin - AWS' section of the Rancher interface. It displays a list of permissions for a user account. The 'Global Permissions' section includes 'Administrator' (unchecked), 'Standard User' (checked), and 'User-Base' (unchecked). The 'Custom' section is empty. The 'Built-in' section lists several permissions: 'Create new Clusters' (unchecked), 'Create RKE Template Revisions' (unchecked), 'Create new RKE Cluster Templates' (unchecked), 'Configure Authentication' (unchecked), 'Configure Catalogs' (unchecked), and 'Create new Cluster Drivers' (unchecked).

## 11. Now after creating a new user let's see how to add a cluster and provision it from GKE.

- Go to Cluster > Add Cluster

The screenshot shows the 'Add Cluster - Select Cluster Type' page. At the top, there are two main options: 'From existing nodes (Custom)' (with a gear icon) and 'Import an existing cluster' (with an upward arrow icon). Below these, under 'With RKE and new nodes in an infrastructure provider', are icons for Amazon EC2, Azure, DigitalOcean, Linode, and vSphere. Under 'With a hosted Kubernetes provider', are icons for Amazon EKS, Azure AKS, and Google GKE. A 'Cancel' button is at the bottom right.

- Select Google GKE, enter Cluster Name, Add Member (if needed).

Add Cluster - Google GKE

Cluster Name \* awsm-cluster

Member Roles

Name	Role
Default Admin (admin)	Cluster Owner
std-user-1	Member

+ Add Member

Labels & Annotations None

Service Account \*

Service account: private key JSON file

Read from a file

- Next, we need a JSON private key: GCP > IAM & Admin > Service Account > Create key > JSON under the project which was solely allocated for Kubernetes.

Service accounts for project "Awsm Deployment GKE Cluster"

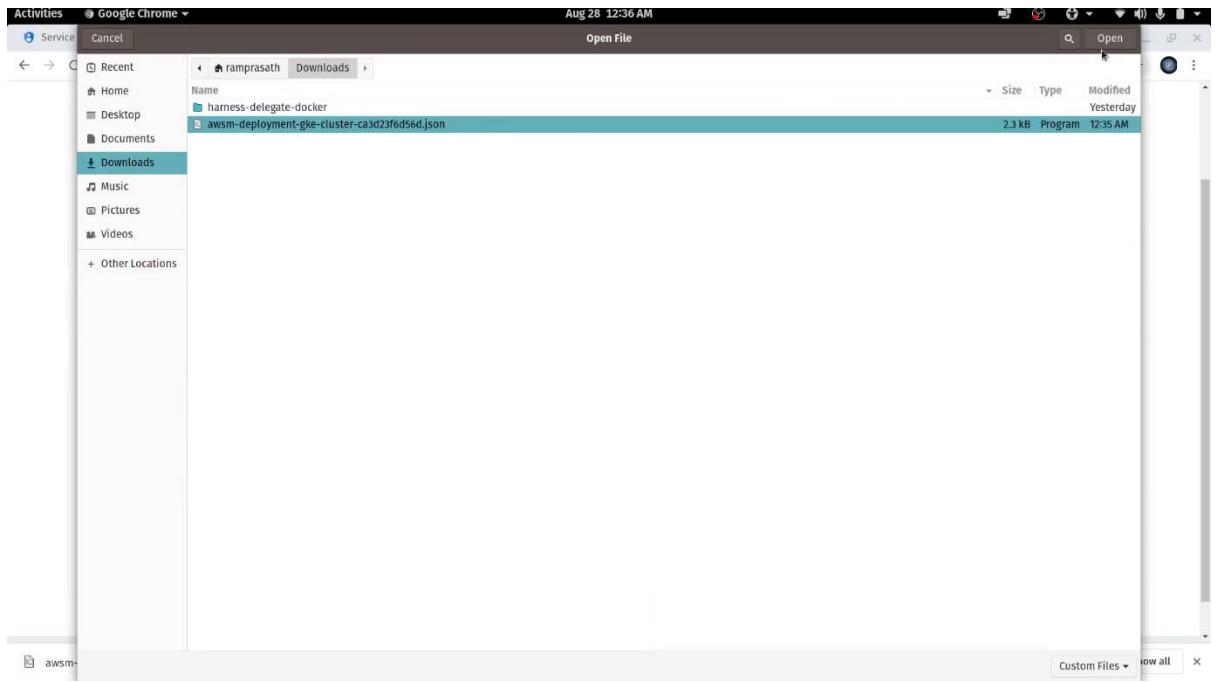
Email	Status	Name ↑	Description	Key ID	Key creation date	Actions
compute@developer.gserviceaccount.com	OK	compute@developer.gserviceaccount.com	Compute Engine default service account	No keys		<span>⋮</span>

Actions: Edit, Disable, Create key, Delete

The screenshot shows the Google Cloud Platform IAM & Admin Service Accounts page. A modal window titled "Create private key for 'Compute Engine default service account'" is open. It contains instructions to download a private key file, a "Key type" section with "JSON" selected (radio button), and a "P12" option. At the bottom are "CANCEL" and "CREATE" buttons.

- After creating the key, we need to upload the key to Rancher.

The screenshot shows the Rancher cluster creation form, step 2. It includes fields for Cluster Name (awsm-cluster), Member Roles (Default Admin (admin) assigned as Cluster Owner, std-user-1 assigned as Member), Labels & Annotations (None), and a Service Account section. The Service Account field has a "Service account private key JSON file" input and a "Read from a file" button. Below the input is a note about required IAM roles: Compute Viewer, Project Editor, Kubernetes Engine Admin, and Service Account User. At the bottom are "Next: Configure Nodes" and "Cancel" buttons.



- Now Rancher will throw an error.

A screenshot of the Rancher UI showing a service account creation page. The page is titled "Create Cluster" and shows a "Service Account" section. A large red warning box is present, stating: "googleapi: Error 403: Identity and Access Management (IAM) API has not been used in project 417265485304 before or it is disabled. Enable it by visiting https://console.developers.google.com/apis/api/iam.googleapis.com/overview?project=417265485304 then retry. If you enabled this API recently, wait a few minutes for the action to propagate to our systems and retry., accessNotConfigured". The Rancher UI interface includes tabs for "Service accounts - IAM &amp; Admin", "Rancher", and "Quotas - IAM &amp; Admin - Aws".

- To resolve the error, we need to enable IAM API in GCP's API Console.

The screenshot shows the 'Identity and Access Management (IAM) API' page in the Google Cloud API Console. At the top, there is a 'TRY THIS API' button and a link to 'Click to enable this API'. Below this, there are tabs for 'OVERVIEW' and 'DOCUMENTATION'. The 'OVERVIEW' tab is selected. On the left, there is a sidebar with a 'About Google' section and a 'Google mission' paragraph. On the right, there is an 'Additional details' section with information about the type, last updated date, and service name. A file download button for 'awsm-deployment.json' is visible at the bottom left, and a 'Show all' button is at the bottom right.

- After enabling the IAM API, an error will be thrown by Rancher to enable another API.

The screenshot shows the Rancher UI for creating a cluster. In the 'Service Account' field, a JSON key is pasted. A warning message in a red box states: 'googleapi: Error 403: Kubernetes Engine API has not been used in project 417265485304 before or it is disabled. Enable it by visiting https://console.developers.google.com/apis/api/container.googleapis.com/overview?project=417265485304 then retry. If you enabled this API recently, wait a few minutes for the action to propagate to our systems and retry. accessNotConfigured'. Below the warning, there are 'Next: Configure Nodes' and 'Cancel' buttons. The bottom navigation bar includes links for Help & Docs, Forums, Slack, and File an Issue.

- To resolve the error, we need to enable Kubernetes Engine API in GCP's API Console.

The screenshot shows the Google Cloud Platform API Console. The URL is [console.developers.google.com/apis/library/container.googleapis.com?project=awsm-deployment-gke-cluster](https://console.developers.google.com/apis/library/container.googleapis.com?project=awsm-deployment-gke-cluster). The page displays information about the Kubernetes Engine API, including its logo, a brief description, and an 'ENABLE' button. Below the main content are tabs for 'OVERVIEW', 'PRICING', and 'DOCUMENTATION'. The 'OVERVIEW' tab is selected. On the right side, there is a sidebar with 'Additional details' such as Type: APIs & services, Last updated: 12/10/19, Category: Compute, and Service name: container.googleapis.com.

- Once the API is enabled, head back to Rancher and click **Configure Nodes**

The screenshot shows the Rancher interface. The URL is [35.240.188.9/g/clusters/add/launch/gogogke](https://35.240.188.9/g/clusters/add/launch/gogogke). The page is titled 'Add Cluster' and shows the 'INTEGRATE WITH RANCHER' section. It lists a 'Default Admin (admin)' user under 'Name' and 'Cluster Owner' under 'Role'. A 'std-user-1' user is listed under 'Member'. There is a '+ Add Member' button. Below this is a 'Labels & Annotations' section with a note about creating a service account. The 'Service Account' section contains a large redacted JSON key. At the bottom, there is an 'Authenticating...' progress bar and a 'Cancel' button. The footer includes links for Help & Docs, Forums, Slack, and File an Issue, along with language and download options.

- Select a desirable zone as per requirements.

The screenshot shows the 'Kubernetes Options' configuration page in the Rancher UI. Key settings visible include:

- Location Type:** Zonal (selected)
- Zone:** us-west1-b
- Additional Zones:** us-west1-a, us-west1-c (checkboxes)
- Kubernetes Version:** 1.16.13-gke.1
- Container Address Range:** e.g. 10.42.0.0/16
- Alpha Features:** Disabled (selected)
- Legacy Authorization:** Disabled (selected)
- Stackdriver Logging:** Enabled (selected)
- Stackdriver Monitoring:** Enabled (selected)
- Kubernetes Dashboard:** Enabled (selected)
- Http Load Balancing:** Enabled (selected)

- Then configure node options

The screenshot shows the 'Node Options' configuration page in the Rancher UI. Key settings visible include:

- Node Count:** 4
- Machine Type:** n1-standard-2 (2 vCPUs, 7.5 GB RAM)
- Image Type:** Container-Optimized OS
- Root Disk Size:** 100 GB
- Local SSD disks:** 0 GB
- Preemptible nodes (beta):** Disabled (selected)
- Auto Upgrade:** Disabled (selected)

- Scroll down and click **Create**. The provisioning of our new cluster will take about 2-5 minutes.

Cluster Name	Provider	Nodes	CPU	RAM
awsm-cluster	Google GKE	0	n/a	n/a

- Once the status of the cluster is active as displayed below. Then, we have successfully created a cluster using GKE.

Cluster Name	Provider	Nodes	CPU	RAM
awsm-cluster	Google GKE	4	n/a	n/a

## 12. Once we have created the cluster, let's enable monitoring

- Clusters > awsm-cluster > Tools > Monitoring

Dashboard: awsm-cluster

Provider: Google GKE (us-west1-b) | Kubernetes Version: v1.16

Created: 12:38 AM

Enable Monitoring to see live metrics

CPU: 20% (1.5 of 7.7 Reserved)

Memory: 7% (1.5 of 22 GiB Reserved)

Pods: 7% (29 of 440 Used)

Etcd, Controller Manager, Scheduler, Nodes

Events

<https://35.240.188.90/c/c-zmbfm/monitoring/cluster-setting>

- Enable Persistent Storage for Prometheus for actual use case.

Data Retention: 12 hours

Enable Node Exporter: True

Enable Persistent Storage for Prometheus: False

Prometheus CPU Limit: 1000 milli CPUs

Prometheus Memory Limit: 1000 MiB

Prometheus CPU Reservation: 750 milli CPUs

Prometheus Memory Reservation: 750 MiB

Node Exporter CPU Limit: 200 milli CPUs

Node Exporter Memory Limit: 200 MiB

Node Exporter Host Port: 9796

Prometheus Operator Memory Limit: 500 MiB

Select the nodes where monitoring related workloads will be scheduled to:

+ Add Selector

Tolerations for Prometheus pod:

+ Add Tolerance

Enable Persistent Storage for Grafana:

True (radio button selected)

Show advanced options

- Then scroll down and click **Enable**.

Prometheus CPU Limit: 1000 milli CPUs

Prometheus Memory Limit: 1000 MiB

Prometheus CPU Reservation: 750 milli CPUs

Prometheus Memory Reservation: 750 MiB

Node Exporter CPU Limit: 200 milli CPUs

Node Exporter Memory Limit: 200 MiB

Node Exporter Host Port: 9796

Prometheus Operator Memory Limit: 500 MiB

Select the nodes where monitoring related workloads will be scheduled to:

+ Add Selector

Tolerations for Prometheus pod:

+ Add Tolerance

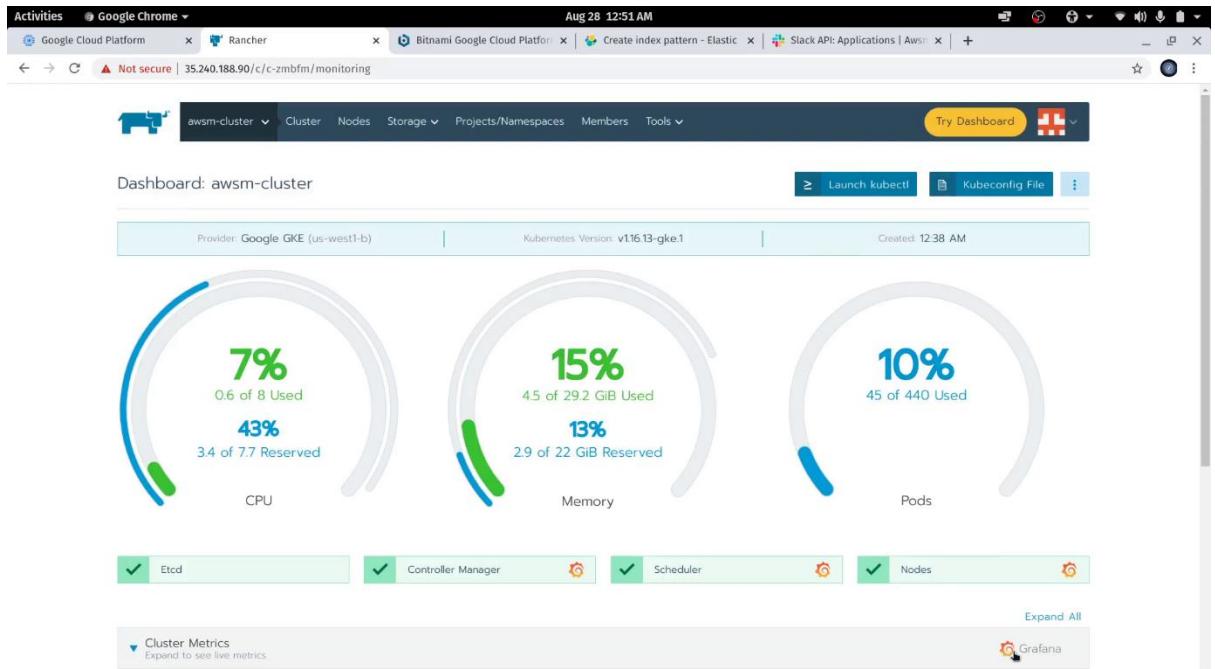
Enable Persistent Storage for Grafana:

True  False

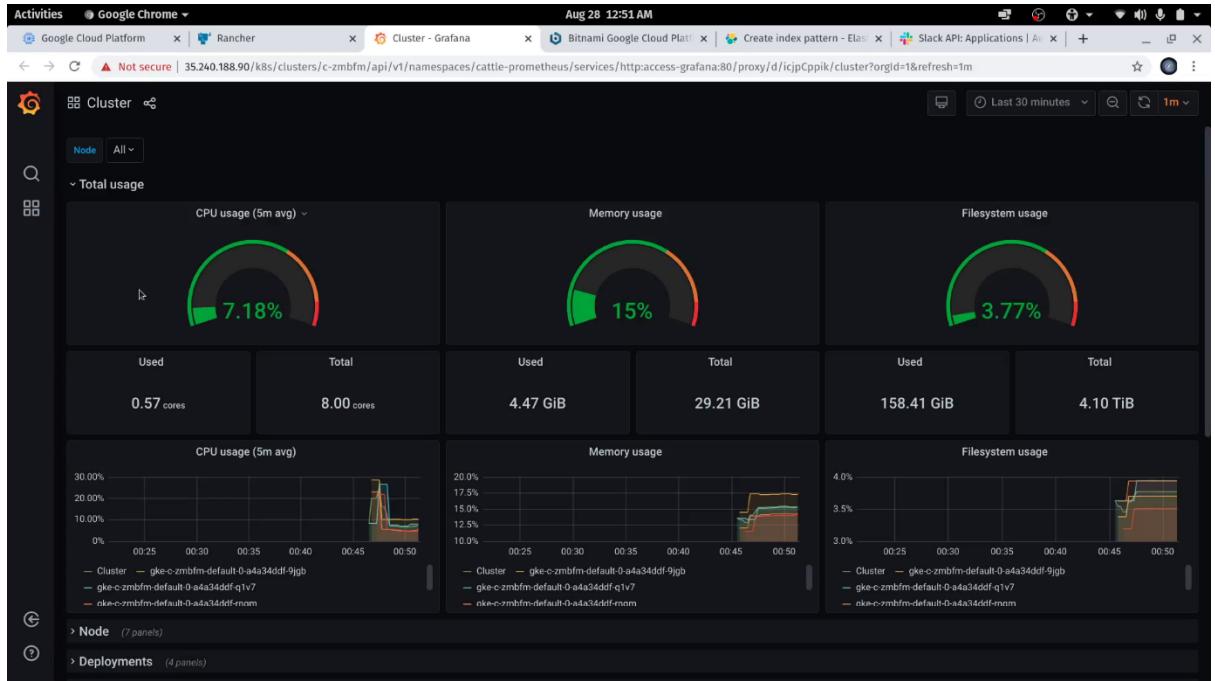
Show advanced options

Enable

- Once monitoring is enabled and the monitoring API is ready, we can go to Grafana dashboard by clicking on the Grafana icon.



- We can now access various metrics from Grafana dashboard.

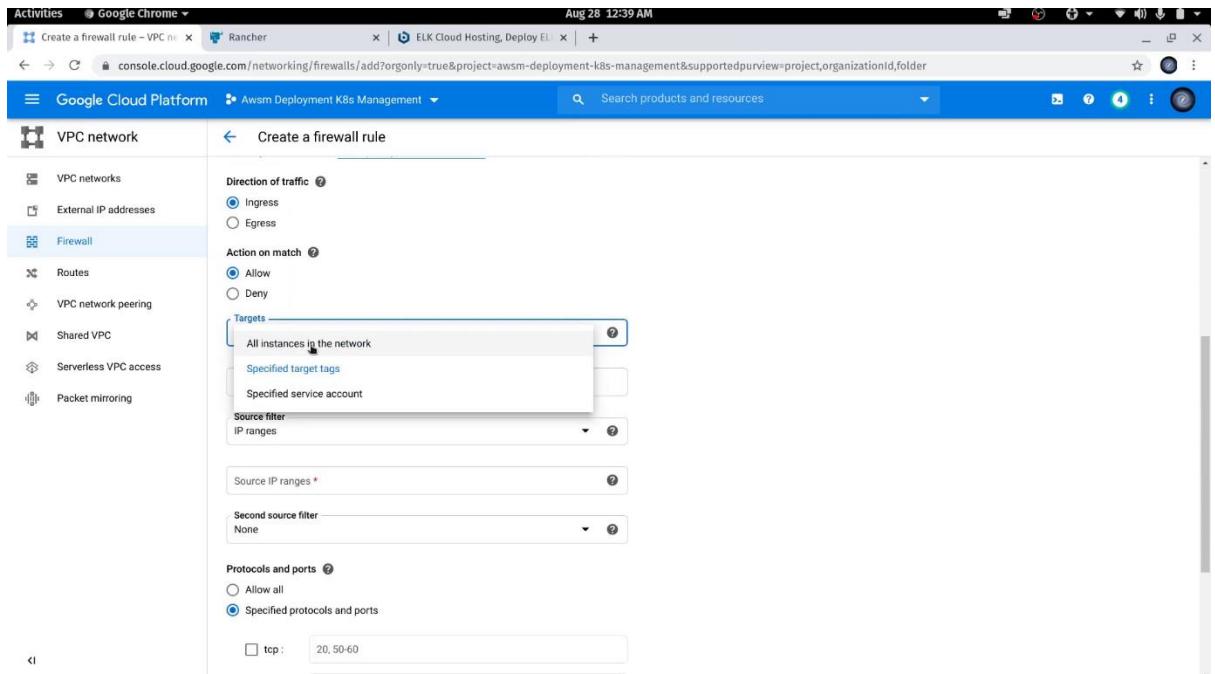


### 13. Now we will setup ELK Stack using Bitnami Launchpad for Google Cloud Platform.

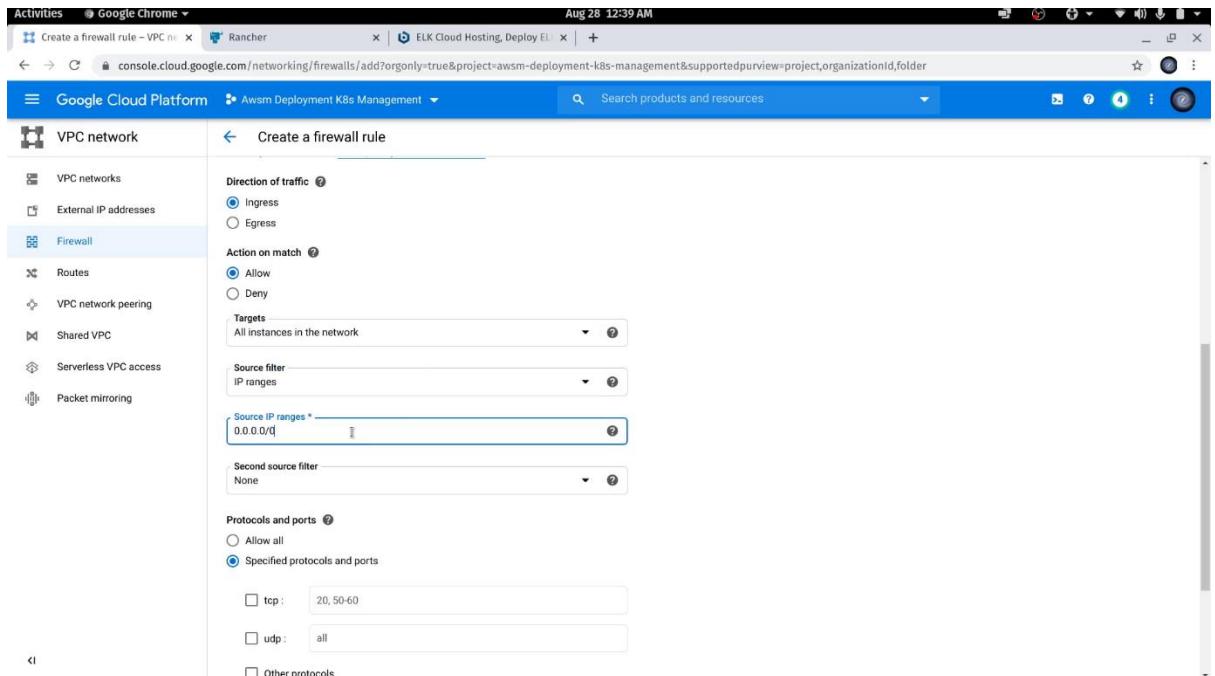
- Note: Before going to the link we need to add a firewall rule in GCP to allow TCP port 9200 for Elasticsearch.

The screenshot shows the "Create a firewall rule" page in the Google Cloud Platform VPC network section. The "Name" field is set to "awsrm-elasticsearch". The "Logs" section has "Off" selected. The "Priority" is set to "1000". The "Direction of traffic" is "Ingress". The "Action on match" is "Allow".

- Since this is a walkthrough, I am choosing **all instances in the network** but for actual use case we need to choose **specified target tags** (here, tag of ELK stack instance).



- For source IP ranges, enter **0.0.0.0/0**



- Then select **tcp**, add port number **9200** and click **Create**.

- Now we will head to Bitnami's cloud ELK Stack using the following link: <https://bitnami.com/stack/elk/cloud>
- Sign in to Bitnami with your google account which is used for accessing GCP account and launch ELK.

here. If you continue to use this site, you consent to our use of cookies.'"/&gt;

- **Unlock Bitnami Vault**

**Unlock Your Bitnami Vault**

This password is independent from your Google Cloud Platform account and primary Bitnami account, and is used to secure access to sensitive information such as SSH keys or API credentials.

Vault password

**UNLOCK**

Please click [here](#) if you do not remember your password.

© Bitnami 2020. [Privacy Policy](#) [Your California Privacy Rights](#)

- Then we need to add project which is not used by Kubernetes cluster

**New Virtual Machine**

NAME

IMAGE

CLOUD ACCOUNT  Associated project

We can't connect to this project, please [click here](#) to re-authenticate it.

NETWORK You have no networks configured for this project. Please create one on the developer console before continuing.

DISK TYPE  Loading available disks

SERVER SIZE  Loading available server sizes

REGION  No regions available. This may be because your project is still provisioning, please try again in a few minutes.

[https://google.bitnami.com/services/new/setup?image\\_id=cWNHPgY\\_ost](https://google.bitnami.com/services/new/setup?image_id=cWNHPgY_ost):

- Choose the appropriate project

**Project setup**

Sign in with Google  
RAMPRASATHASOKAN@GMAIL...

**Select your project**

Enable Compute API

Enable DM API

Enable billing

Finished

STEP 2 OF 6

**Select your project**

In order to launch instances on your behalf, you must first specify which project you would like to use. You can create and manage projects from the [Google Developers Console](#).

Awm Deployment K8s Management (awsm-deployment-k8s-management)

+ New Project    Refresh Projects    3 PROJECTS LOADED

CONTINUE

© Bitnami 2020. [Privacy Policy](#) [Your California Privacy Rights](#)

- Enable Deployment Manager API

**Project setup**

Sign in with Google  
RAMPRASATHASOKAN@GMAIL...

Select your project  
AWSM-DEPLOYMENT-K8S-MA...

Enable Compute API

**Enable DM API**

Enable billing

Finished

STEP 4 OF 6

**Enable Deployment Manager API**

In order to create instances with your project "awsm-deployment-k8s-management" the Google Deployment Manager V2 API must be [enabled from the developer console](#).

The above link may not work if logged in with multiple Google accounts. Complete instructions on enabling Google APIs are [available here](#).

<https://console.developers.google.com/apis/library/deploymentmanager.googleapis.com?project=awsm-deployment-k8s-management>

Google Developers Console

RPI API Manager Overview

Enable API

Google Cloud Deployment Manager V2 API

Using credentials with this API

Accessing user data with OAuth 2.0

Your app

User consent

Activities ● Google Chrome Aug 28 12:41 AM

Google Cloud Platform | Rancher | Bitnami Google Cloud Platfo | Cloud Deployment Manager | +

console.developers.google.com/apis/library/deploymentmanager.googleapis.com?project=awsm-deployment-k8s-management

☰ Google APIs A fsm Deployment K8s Management Search for APIs and Services

Cloud Deployment Manager V2 API

The Google Cloud Deployment Manager v2 API provides services for configuring, deploying,...

ENABLE TRY THIS API Click to enable this API

OVERVIEW DOCUMENTATION

**Overview**

The Google Cloud Deployment Manager v2 API provides services for configuring, deploying, and viewing Google Cloud services and APIs via templates which specify deployments of Cloud resources.

**About Google**

Google's mission is to organize the world's information and make it universally accessible and useful. Through products and platforms like Search, Maps, Gmail, Android, Google Play, Chrome and YouTube, Google plays a meaningful role in the daily lives of billions of people.

Tutorials and documentation

● Then head back to Bitnami and hit **CONTINUE**. Next, click **LAUNCH**.

Activities ● Google Chrome Aug 28 12:41 AM

Google Cloud Platform | Rancher | Bitnami Google Cloud Platfo | Overview – APIs & Services | +

google.bitnami.com/services/WsvhNNc/setup/finish

Bitnami Launchpad for Google Cloud Platform

VIRTUAL MACHINES LIBRARY SUPPORT ACCOUNT

**Project setup**

- Sign in with Google RAMPRASATHASOKAN@GMAIL...
- Select your project AWSM-DEPLOYMENT-K8S-MA...
- Enable Compute API
- Enable DM API
- Enable billing
- Finished

STEP 6 OF 6

You're all done!

Launch ELK Now Launch ELK on Google Cloud Platform now. LAUNCH >

© Bitnami 2020. Privacy Policy Your California Privacy Rights

- Now configure instance type, select desirable region as per requirements and click CREATE.

Bitnami Launchpad for Google Cloud Platform

**NETWORK**: default

**DISK TYPE**: Magnetic Disk

**DISK SIZE**: 50 GB

**SERVER SIZE**: n1-standard-1

Estimated Monthly cost: \$36.67

**CANCEL** **CREATE**

© Bitnami 2020. Privacy Policy Your California Privacy Rights

- Then we will be redirected to the console of the ELK instance in Bitnami Launchpad.

bitnami-elk-7fb8

Create virtual machine: Starting operation 0%

**Application Info**

**ELK** 7.9.0-1  
ELK stack combines three open source projects for log management: Elasticsearch as a search and analytics engine, Logstash for centralizing...

**CREDENTIALS**

USERNAME: user

PASSWORD:  SHOW

PORTS: 80, 443

**Server Info**

Not started yet

**N1-STANDARD-1**  
\$34.67/MO (0.047/HR)

**MAGNETIC DISK**  
50 GB (\$2.00/MO)

**US-CENTRAL1-C**  
REGION

**\$36.67**  
ESTIMATED MONTHLY COST

CONNECTION NOT AVAILABLE  
**LAUNCH SSH CONSOLE**

DOWNLOAD KEY: [.PEM](#) [.PPK](#)

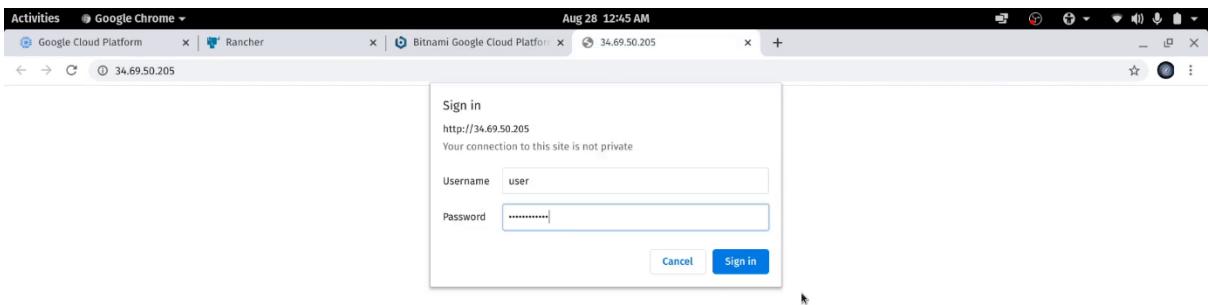
- We can verify the launch of ELK instance from GCP under the respective chosen project.

The screenshot shows the Google Cloud Platform Compute Engine VM Instances page. The left sidebar lists various Compute Engine options like Instance groups, Instance templates, Sole-tenant nodes, Machine images, Disks, Snapshots, Images, TPUs, Migrate for Compute Engi..., Committed use discounts, Metadata, Health checks, Zones, and Marketplace. The main area displays a table of VM instances with columns: Name, Zone, Recommendation, In use by, Internal IP, External IP, and Connect. Two instances are listed: 'awsm-rancher-server' in 'asia-southeast1-b' with Internal IP 10.148.0.2 (nic0) and External IP 35.240.188.90; and 'bitnami-elk-7fb8' in 'us-central1-c' with Internal IP 10.128.0.2 (nic0) and External IP 34.69.50.205. Below the table are related actions: View Billing Report, Monitor VMs, Explore VM Logs, Setup Firewall Rules, and Patch Management.

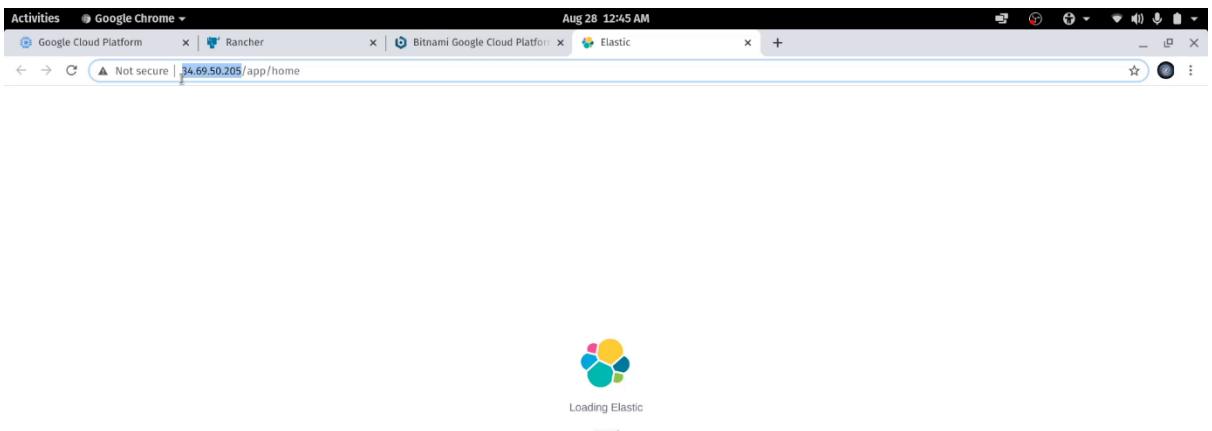
- Once the instance is created, click **GO TO APPLICATION**

The screenshot shows the Bitnami Launchpad for Google Cloud Platform interface. At the top, there's a navigation bar with tabs for VIRTUAL MACHINES, LIBRARY, SUPPORT, and ACCOUNT. The main area is divided into two sections: Application Info and Server Info. The Application Info section features a logo for ELK 7.9.0-1, a description of the ELK stack, and two buttons: 'GO TO APPLICATION' (LAUNCHES IN A NEW WINDOW) and 'GO TO ADMINISTRATION' (LAUNCHES IN A NEW WINDOW). It also includes a 'CREDENTIALS' section with fields for USERNAME (user), PASSWORD (redacted), and PORTS (80, 443). The Server Info section shows the instance is 'Running' (2 minutes ago) with IP 34.69.50.205. It details the server configuration: N1-STANDARD-1 (34.67/MO (0.047/HR)), MAGNETIC DISK (50 GB (\$2.00/MO)), US-CENTRAL1-C REGION, and an estimated monthly cost of \$36.67. It includes a 'LAUNCH SSH CONSOLE' button, download keys for PEM and PPK, and a 'Show SSH command' link. At the bottom, there are buttons for REBOOT, SHUTDOWN, and DELETE.

- Then enter **Username** and **Password** to **Sign in**.



- Now, copy the external ip address of the ELK instance



- Go to **Rancher > Cluster > awsm-cluster > Tools > Logging**
- Note: Make sure that monitoring API is up and Grafana is running before setting up logging.

The screenshot shows the Rancher interface for the awsm-cluster. At the top, there are tabs for 'Tools' (selected), 'Alerts', 'Catalogs', 'Notifiers', 'Monitoring' (highlighted in blue), 'Istio', and 'CIS Scans'. Below this, three circular progress bars show resource usage: CPU at 43% (3.4 of 7.7 Reserved), Memory at 13% (2.9 of 22 GiB Reserved), and Pods at 9% (38 of 440 Used). A note says 'Monitoring API is not ready'. Below the bars is a row of green status indicators for 'Etcfd', 'Controller Manager', 'Scheduler', and 'Nodes'. At the bottom, a section titled 'Events' shows 'Events of current Cluster'. The URL in the address bar is <https://35.240.188.90/c/c-zmbfm/logging>.

- Now select Elasticsearch from the following

The screenshot shows the 'Cluster Logging' configuration page. It includes a note about using fluentd to collect logs from containers and hosts. Below are six options for log collectors: 'None' (selected), 'Elasticsearch' (selected), 'Splunk', 'Kafka', 'Syslog', and 'Fluentd'. A note below says 'Logging is disabled in the current cluster.' A 'Save' button is at the bottom. The URL in the address bar is <https://35.240.188.90/c/c-zmbfm/logging>. The footer includes links for v2.4.6, Help & Docs, Forums, Slack, File an Issue, English, and Download CLI.

- Enter the external ip address of the ELK instance which we copied earlier and include the 9200 port along with it in the Elasticsearch endpoint field.

We will use fluentd to collect stdout/stderr logs from each container and the log files which exist under path `/var/log/containers/` on each host. The logs can be shipped to a target you configure below.

**Edit as File**

No logging target, click the Save button below to set **Elasticsearch** as the logging target.

**Elasticsearch Configuration**

Endpoint \*

`http://34.69.50.205:9200`

Copy your endpoint from Elastic Cloud, or input the reachable endpoint of your self-hosted Elasticsearch.

X-Pack Security (Optional)

- Let's **TEST** whether the given endpoint is accessible, if not, check whether the tcp port 9200 was added to the firewall else create a new firewall rule and add the port.

Custom Log Fields

You can add custom fields as key/value pairs for better filtering.

**+ Add Field**

Flush Interval

60 sec

How often buffered logs would be flushed.

Include System Log

Enable JSON Parsing

```
{
  "_index": "awsm-cluster-2020-08-28",
  "_id": "WD68LUuhwVvf5LMjqlh",
  "_source": {
    "log": {
      "time": "2018-01-15T17:49:26Z",
      "level": "info",
      "msg": "Creating cluster event [Created cluster event]"
    },
    "kubernetes": {
      "container_name": "cattle",
      "namespace_name": "default",
      "pod_name": "cattle-6b4ccb5b9d-tzs4q",
      "labels": {
        "app": "cattle",
        "pod-template-hash": "2607761658"
      },
      "host": "47.89.14.205",
      "master_url": "https://10.233.0.1:443/api"
    }
  }
}
```

**TEST**

**Save**

v2.4.6 Help & Docs Forums Slack File an Issue

English Download CLI

- If the **Settings validated** is displayed, click **Save**.

The screenshot shows the Bitnami Google Cloud Platform interface with the URL <https://35.240.188.90/c/c-zmbfm/logging?targetType=elasticsearch>. The page title is "Additional Logging Configuration". On the left, there's a section for "Custom Log Fields" with a "Add Field" button. Below it is a "Flush Interval" input set to 60 seconds. There are checkboxes for "Include System Log" (checked) and "Enable JSON Parsing". A large text area on the right shows a JSON log entry:

```
{
  "index": "awsm-cluster-2020-08-28",
  "id": "AW68LuuhvVvfSLMjqlh",
  "source": {
    "log": "time=2018-01-15T17:49:26Z level=info msg=Creating cluster event [Created co",
    "kubernetes": {
      "container_name": "cattle",
      "namespace_name": "default",
      "pod_name": "cattle-6b4ccb5b9d-tzs4q",
      "labels": {
        "app": "cattle",
        "pod-template-hash": "2607761658"
      },
      "host": "47.89.14.205",
      "master_url": "https://10.233.0.1:443/api"
    }
  }
}
```

At the bottom right, there is a green "Settings validated" button and a blue "Save" button.

- Now head to the Kibana dashboard, select **Explore on my own**.

The screenshot shows the Kibana dashboard with the URL [https://34.69.50.205/app/home#/. The page features a central graphic of a gear and a bar chart. Below the graphic, the text "Let's get started" is displayed, followed by a message: "We noticed that you don't have any data in your cluster. You can try our sample data and dashboards or jump in with your own data." At the bottom, there are two buttons: "Try our sample data" and "Explore on my own". A small note at the bottom right states: "To learn about how usage data helps us manage and improve our products and services, see our \[Privacy Statement\]\(#\). To stop collection, disable usage data here."](https://34.69.50.205/app/home#/)

- Navigate to Discover.

The screenshot shows the Bitnami Google Cloud Platform interface with the Kibana Discover section selected in the left sidebar. The main area displays various data sources and options for interacting with Elasticsearch data. A prominent 'Discover' button is visible, along with sections for 'Add sample data', 'Use Elasticsearch data', and 'Manage and Administer the Elastic Stack'.

- Then, Create index pattern

The screenshot shows the Bitnami Google Cloud Platform interface with the Index patterns section selected in the left sidebar. A message at the top indicates that an index pattern needs to be created to visualize and explore data. The main area displays an 'Index patterns' search bar and a list that shows 'No items found'.

- Next, Define index pattern

The screenshot shows the 'Create index pattern' page in Kibana. The title is 'Create index pattern'. A sub-header says 'An index pattern can match a single source, for example, filebeat-4-3-22, or multiple data sources, filebeat-\*.' Below this is a 'Read documentation' link. The main section is titled 'Step 1 of 2: Define index pattern'. It has a form field labeled 'Index pattern name' containing 'awsm-cluster-2020-08-27'. A note below it says 'Use an asterisk (\*) to match multiple indices. Spaces and the characters \, /, ?, \*, <, >, | are not allowed.' There is a checkbox 'Include system and hidden indices' which is unchecked. A message box at the bottom says 'Your index pattern matches 1 source.' Below the form is a table with one row labeled 'awsm-cluster-2020-08-27' and an 'Index' column. At the bottom right is a 'Next step >' button.

- Configure settings: Time field > @timestamp

The screenshot shows the 'Create index pattern' page in Kibana, specifically Step 2 of 2: Configure settings. The title is 'Create index pattern'. A sub-header says 'An index pattern can match a single source, for example, filebeat-4-3-22, or multiple data sources, filebeat-\*.' Below this is a 'Read documentation' link. The main section is titled 'Step 2 of 2: Configure settings'. It has a form field labeled 'Time field' with a dropdown menu open. The menu contains '@timestamp' (which is highlighted with a blue selection bar) and 'I don't want to use the Time Filter'. At the bottom right are 'Back' and 'Create index pattern' buttons.

- Then hit **Create index pattern**.

Kibana

**Index Patterns**

Saved Objects

Advanced Settings

Create index pattern

Create index pattern

An Index pattern can match a single source, for example, `filebeat-4-3-22`, or **multiple** data sources, `filebeat-*`. [Read documentation](#)

**Step 2 of 2: Configure settings**

**awsrm-cluster-2020-08-27**

Select a primary time field for use with the global time filter.

Time field Refresh  
@timestamp

Show advanced options

< Back Create index pattern

- Now, head back to **Discover**, we can see various logs with their respective timestamp.

Discover

New Save Open Share Inspect

Search

KQL Last 15 minutes Show dates Refresh

awsrm-cluster-2020-08-27

Count

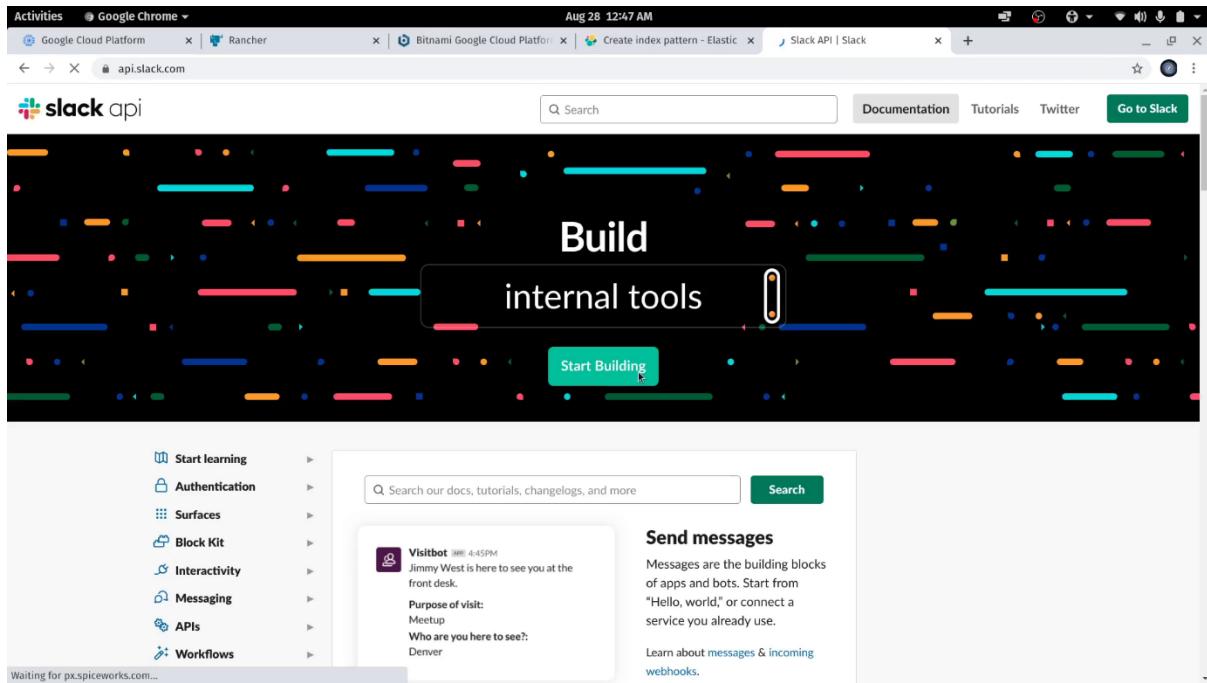
Aug 28, 2020 @ 00:38:25.861 - Aug 28, 2020 @ 00:53:25.861 Auto

Time	_source
Aug 28, 2020 @ 00:52:46.386	log: level:info ts=2020-08-27T19:22:46.385144713Z caller=operator.go:1094 component=prometheusoperator msg="sync prometheus" key=cattle-prometheus/cluster-monitoring stream: stdout docker.container_id: 62ed6a16bdae22e5b1e8ae847c7f6fffb982c5debf29224c161e446ab6c3d828f kubernetes.container_name: prometheus-operator kubernetes.namespace_name: cattle-prometheus kubernetes.pod_name: prometheus-operator-monitoring-operator-5f5d4c5987-d5gxb kubernetes.container_image: rancher/coreos-prometheus-operator:v0.38.1 kubernetes.container_image_id: docker-pullable://rancher/coreos-prometheus-
Aug 28, 2020 @ 00:52:37.401	log: level:info ts=2020-08-27T19:22:37.399521671Z caller=operator.go:1094 component=prometheusoperator msg="sync prometheus" key=cattle-prometheus/cluster-monitoring stream: stdout docker.container_id: 62ed6a16bdae22e5b1e8ae847c7f6fffb982c5debf29224c161e446ab6c3d828f kubernetes.container_name: prometheus-operator kubernetes.namespace_name: cattle-prometheus kubernetes.pod_name: prometheus-operator-monitoring-operator-5f5d4c5987-d5gxb kubernetes.container_image: rancher/coreos-prometheus-operator:v0.38.1 kubernetes.container_image_id: docker-pullable://rancher/coreos-prometheus-

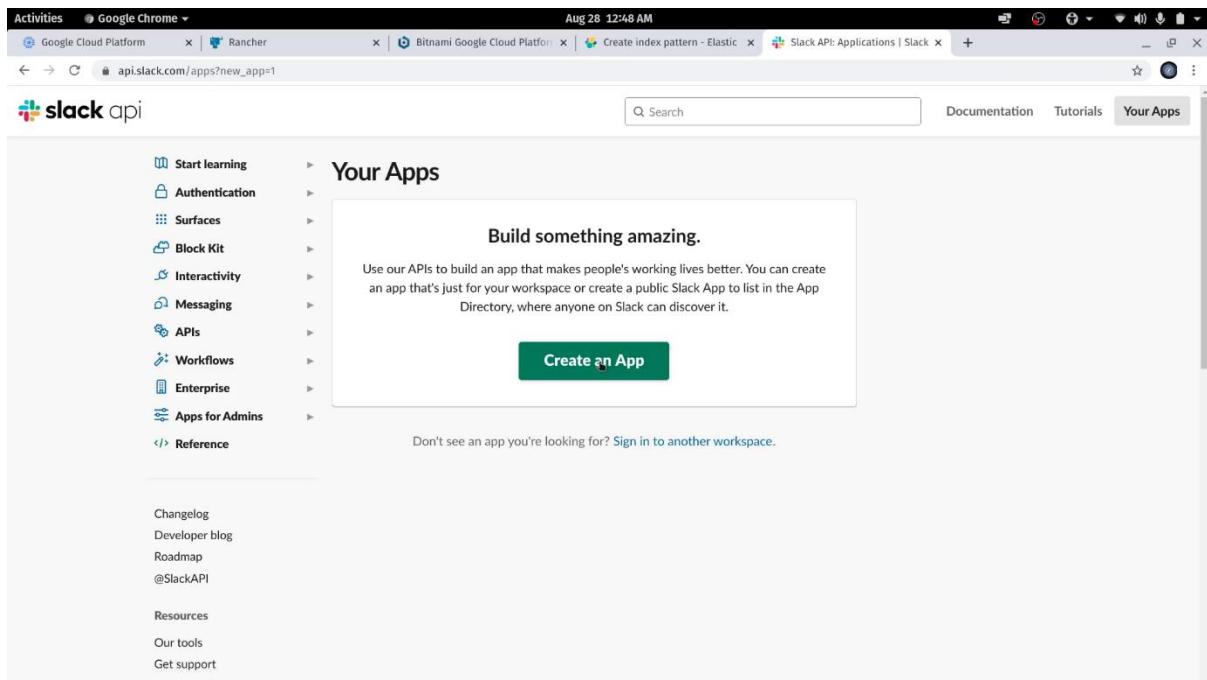
- We have now successfully configured ELK stack with our Kubernetes cluster.

14. Now we will setup Slack for notifications regarding our cluster alerts.

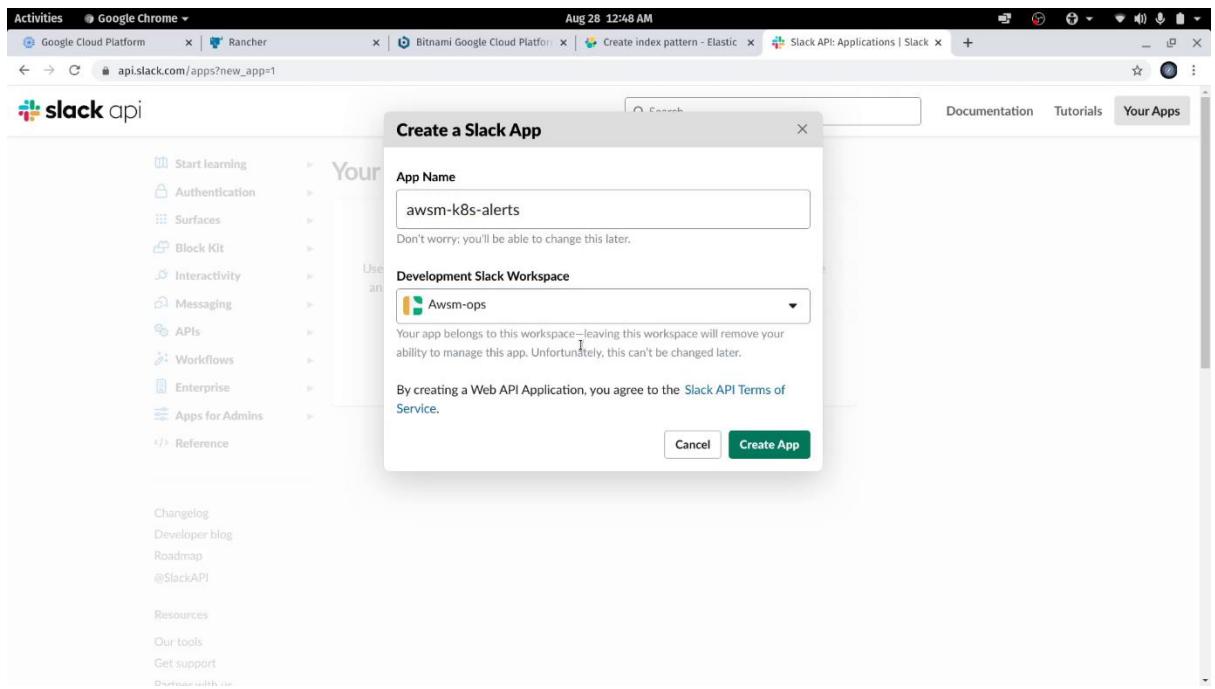
- Create a Slack workspace.
- Create a channel.
- Then go to <https://api.slack.com> and click **Start Building**.



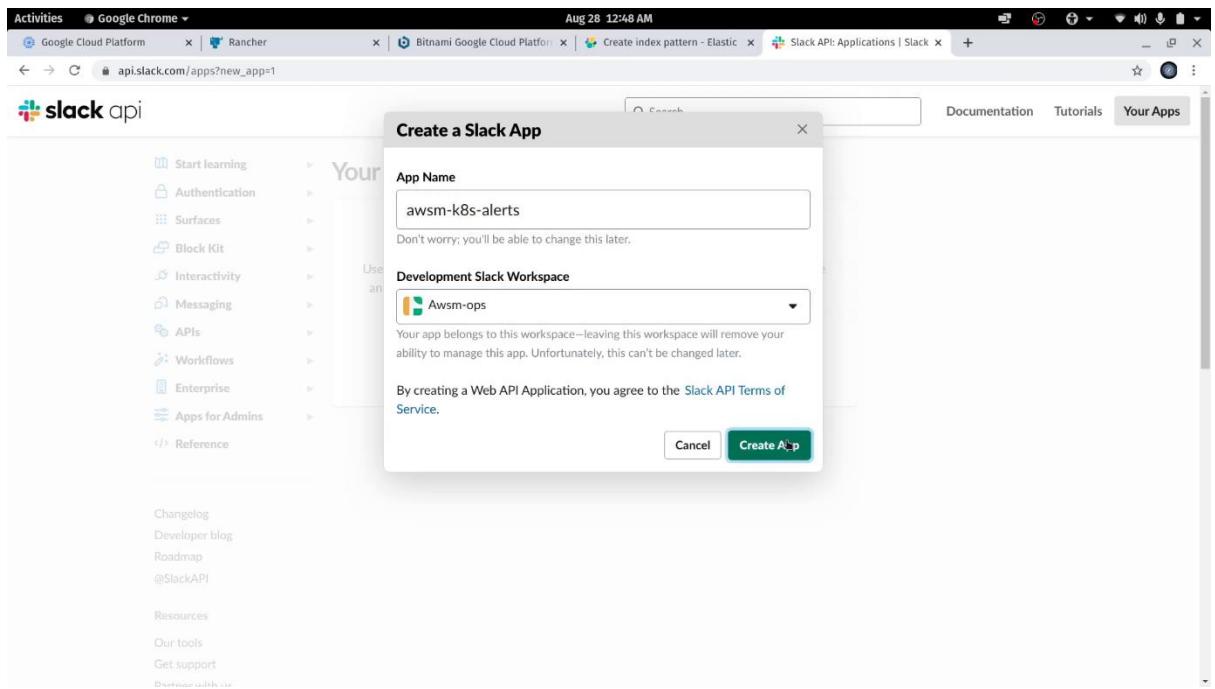
- Click on **Create an App**.



- Enter an App Name and select a Development Slack Workspace.



- Then click Create App.



- Click Incoming Webhooks under Features.

The screenshot shows the Slack API application configuration interface. The left sidebar has a 'Features' section with 'Incoming Webhooks' selected. The main content area is titled 'Basic Information' and contains sections for 'Building Apps for Slack' and 'Add features and functionality'. Under 'Add features and functionality', there are four options: 'Incoming Webhooks' (selected), 'Interactive Components', 'Slash Commands', and 'Event Subscriptions'. At the bottom right are 'Discard Changes' and 'Save Changes' buttons.

- Activate Incoming Webhooks

The screenshot shows the 'Incoming Webhooks' configuration page. The left sidebar has 'Incoming Webhooks' selected. The main content area is titled 'Activate Incoming Webhooks' and includes a description of what webhooks are, a note about needing a bot user, and a message about generating a new Webhook URL. A toggle switch labeled 'Off' is shown. At the bottom right are 'Discard Changes' and 'Save Changes' buttons.

- Once Incoming Webhooks are activated, proceed to the next step.

The screenshot shows the Slack API Applications interface. The left sidebar has a tree view with 'Incoming Webhooks' selected under 'Features'. The main content area is titled 'Incoming Webhooks' and contains the following sections:

- Activate Incoming Webhooks:** A toggle switch is set to 'On'. Below it, text explains that incoming webhooks are simple HTTP requests with a JSON payload containing the message and optional details. It also notes that adding a bot user is required if none exists.
- Webhook URLs for Your Workspace:** Instructions for dispatching messages via a curl command. The sample command is:
 

```
curl -X POST -H 'Content-type: application/json' --data '{"text":"Hello, World!"}' YOUR_WEBHOOK_URL_HERE
```

- Add New Webhook to Workspace

The screenshot shows the same Slack API Applications interface as the previous one, but with a new webhook added. The main content area now includes a table for managing webhooks:

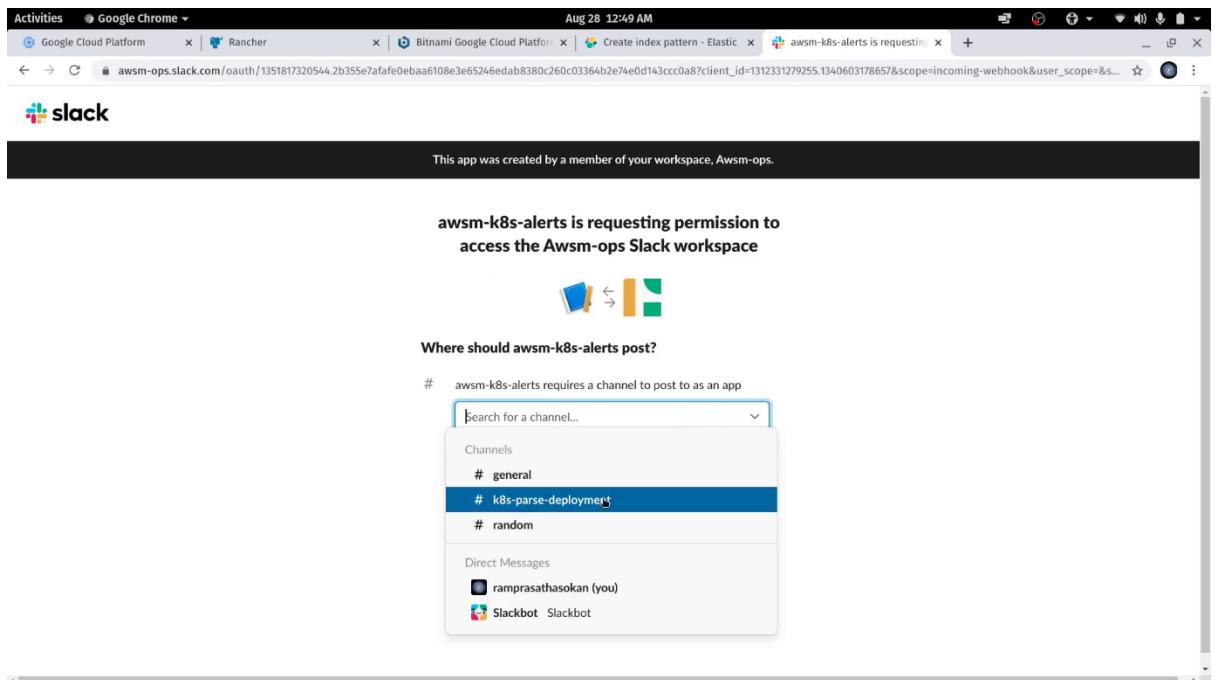
Webhook URL	Channel	Added By
No webhooks have been added yet.		

At the bottom of the table is a button labeled 'Add New Webhook to Workspace'.

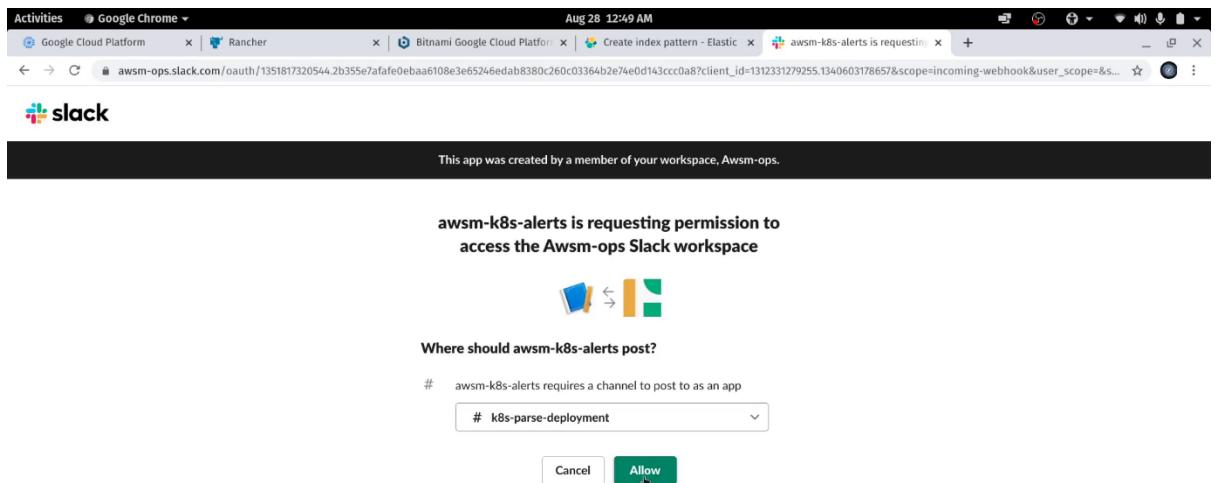
A URL is visible at the bottom of the browser window:

[https://slack.com/oauth/v2/authorize?client\\_id=1312331279255.1340603178657&tea...](https://slack.com/oauth/v2/authorize?client_id=1312331279255.1340603178657&tea...)

- Select for a channel to post alerts.



- Now click **Allow**.



- Once the Webhook is created, copy the Webhook URL.

The screenshot shows a Google Chrome browser window with multiple tabs open. The active tab is 'api.slack.com/apps/A01A0HR58KB/incoming-webhooks?success=1'. The page displays a success message: 'When your app is installed to your team, if you'd like to remove access to existing Webhook URLs, you will need to Revoke All OAuth Tokens.' Below this, there's a section titled 'Webhook URLs for Your Workspace' with instructions to dispatch messages using a curl command. A sample curl request is provided:

```
curl -X POST -H 'Content-type: application/json' --data '{"text":"Hello, World!"}' https://hooks.slack.com/services/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
```

A 'Copy' button is available for the URL. Below the curl example is a table showing the created webhook:

Webhook URL	Channel	Added By
<a href="https://hooks.slack.com/services/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">https://hooks.slack.com/services/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</a>	#k8s-parse-deployment	ramprasathasokan Aug 27, 2020

At the bottom, there's a 'Add New Webhook to Workspace' button.

- Now let's head to Rancher > Cluster > awsm-cluster > Tools > Notifiers

The screenshot shows the Rancher dashboard for the 'awsm-cluster'. The top navigation bar includes 'awscluster', 'Cluster', 'Nodes', 'Storage', 'Projects/Namespace', 'Members', 'Tools' (with 'Notifiers' selected), 'Try Dashboard', and 'Kubeconfig File'. The main area is titled 'Dashboard: awsm-cluster' and shows cluster statistics: Provider: Google GKE (us-west1-b) and Kubernetes Version: v1.16. Three circular progress bars indicate resource usage: CPU (43%, 3.4 of 7.7 Reserved), Memory (13%, 2.9 of 22 GiB Reserved), and Pods (10%, 45 of 440 Used). Below these are status indicators for Etc, Controller Manager, Scheduler, and Nodes, all marked as green with checkmarks. At the bottom, there's a 'Events' section with a note: 'Events of current Cluster'.

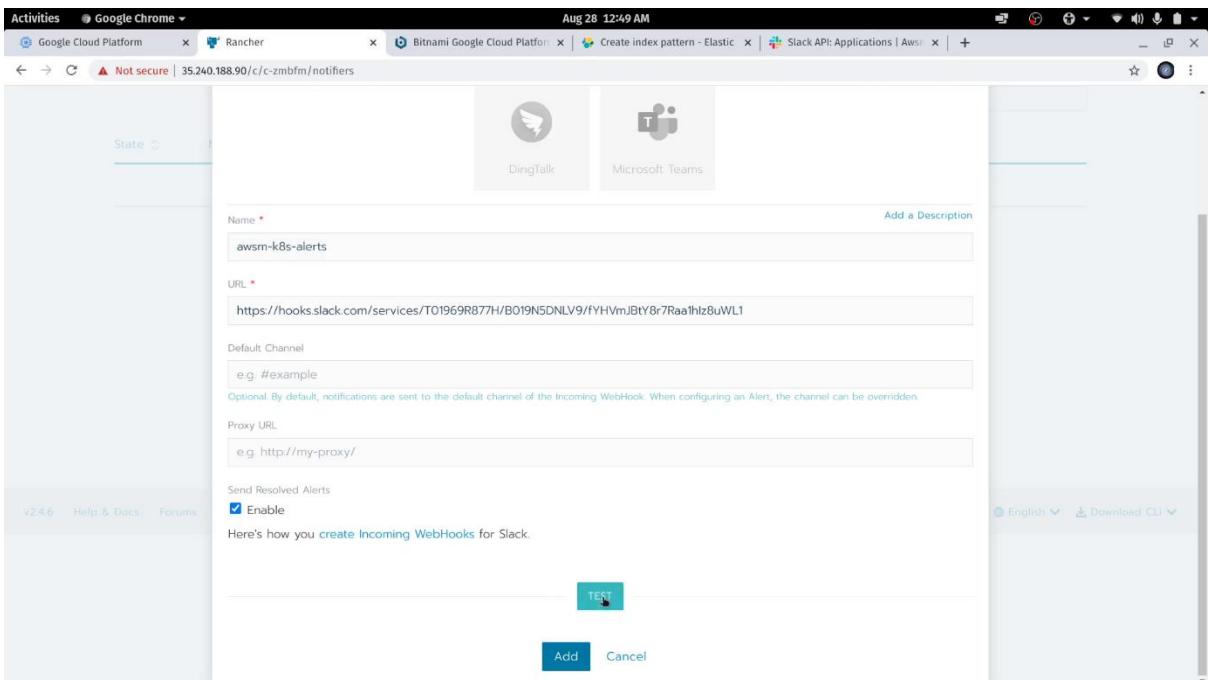
- Click Add Notifier

The screenshot shows the Rancher interface for managing notifiers. At the top, there's a navigation bar with tabs like 'Cluster', 'Nodes', 'Storage', 'Projects/Namespace', 'Members', and 'Tools'. Below the navigation is a sub-navigation for 'Notifier' specific to the 'awsim-cluster'. On the left, there's a sidebar with a 'Notifiers' section. The main content area has a heading 'Notifiers' and a search bar. A prominent button on the right says 'Add Notifier'. Below the button, a message states 'There are no notifiers defined'.

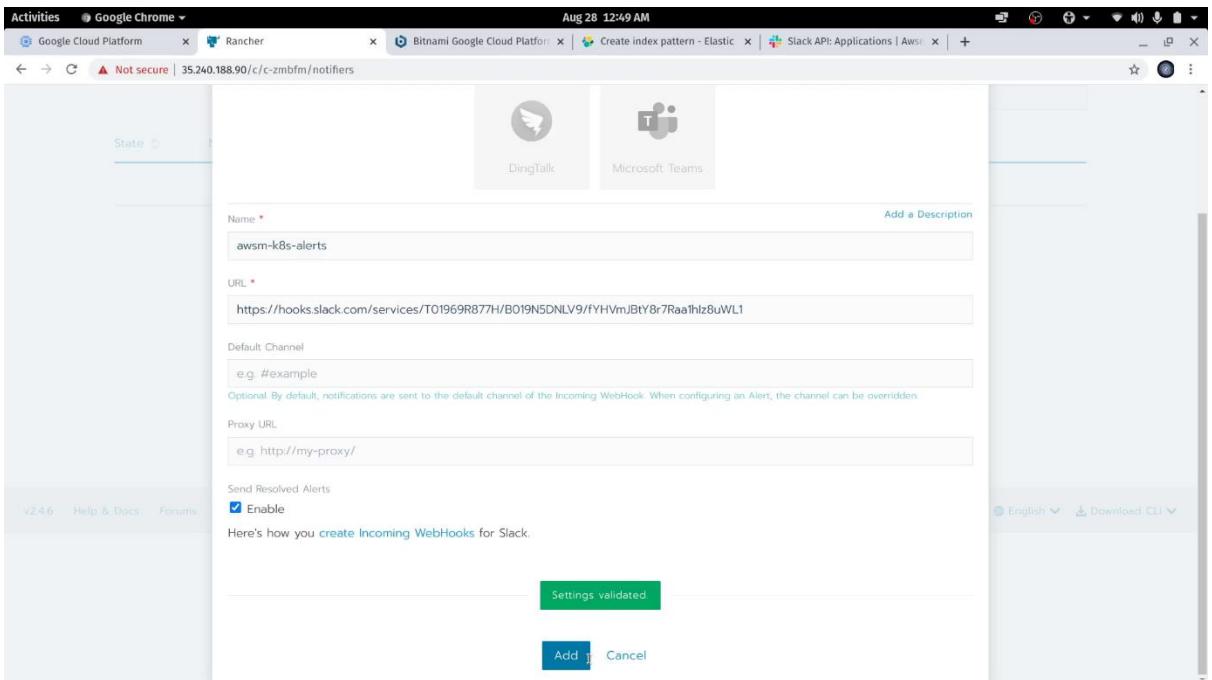
- Enter Webhook URL under **URL** and a **Name**.

This screenshot shows the 'Add Notifier' form for a 'Webhook'. The 'Name' field is populated with 'awsm-k8s-alerts'. The 'URL' field contains the URL 'https://hooks.slack.com/services/T01969R877H/B019N5DNLV9/fyHVmJBy8r7RaIhz8uWL1'. Other fields shown include 'Default Channel' (with placeholder 'e.g. #example') and 'Proxy URL' (with placeholder 'e.g. http://my-proxy/'). There's also a checkbox for 'Send Resolved Alerts' which is unchecked. The right side of the screen shows a sidebar with icons for Slack, Email, PagerDuty, Webhook, WeChat, DingTalk, and Microsoft Teams, along with a 'Card' icon and a 'Add Notifier' button.

- Let's **TEST** whether it's working.



- If **Settings validated** is displayed, click **Add**.
- Note: The default channel is the channel linked to the incoming webhook.



- We can now see that our Slack notifier has been successfully added.

Notifiers

State	Name	Type	Created
Active	awsm-k8s-alerts	Slack Default Channel	a few seconds ago

- Verification from Slack workspace

#k8s-parse-deployment

This is the very beginning of the #k8s-parse-deployment channel  
This channel is for working on a project. Hold meetings, share docs, and make decisions together with your team. [Edit description](#)

Yesterday

ramprasathasokan 7:08 PM joined #k8s-parse-deployment.

Today

ramprasathasokan 12:49 AM added an integration to this channel: awsm-k8s-alerts

awsm-k8s-alerts 12:49 AM Slack setting validated

Hello, team! First order of business...

Send a message to #k8s-parse-deployment

With the Slack app, your team is never more than a click away. [Get Slack for Linux](#) (Already have the app? [Open Slack](#))

- Next, we need to choose the alerts to be notified of. To do so, go to **Rancher > Cluster > awsm-cluster > Tools > Alerts**

The screenshot shows the Rancher interface with the 'Tools' dropdown open, specifically the 'Alerts' tab. The 'Notifiers' section displays a table with one row. The columns are 'State', 'Name', 'Type', and 'Created'. The 'Name' column shows 'awsm-k8s-alerts', the 'Type' column shows 'Slack', and the 'Created' column shows 'a few seconds ago'.

- Then select an alert group to be notified of and click **Edit**.

The screenshot shows the 'Edit Alert Rule' screen for a specific alert group. The alert rule is titled 'Scheduler is unavailable' and is set to trigger on 'System Service' with the condition 'Unhealthy'. The 'Notifiers' field is currently 'Not Configured'. Below this, there are three other alert rules: 'Get warning deployment event', 'High cpu load', and 'Node disk is running full within...'. Each rule has its own edit, add, and delete buttons. At the bottom, there are sections for 'A set of alerts when event happened' and 'A set of alerts for cluster scans'.

- Now choose our newly added Slack notifier.

The screenshot shows a web browser window with multiple tabs open. The active tab is titled 'Create index pattern - Elastic' and displays a chart with numerical values from 0 down to -50000000000. Below the chart, there are time markers: 00:49 08-27, 12:00 08-27, 00:00 08-28, 12:00 08-28, and 00:00 08-29. A legend at the bottom indicates 'Send a Critical' alert (blue dot). Below the chart, it says 'Group Inherited' and 'Enabled'. At the bottom right of the main area is a blue button labeled '+ Add Alert Rule'.

Below this, the 'Alert' section shows 'To' set to 'Choose a Notifier' with a dropdown menu containing 'awsml-k8s-alerts (Slack)' selected. To the right, 'Not Configured' is shown with a note: 'The value set in the Notifier is used by default. You can override this with another value.' A 'Show advanced options' link is visible. At the bottom are 'Save' and 'Cancel' buttons.

The footer of the page includes links for 'v2.4.6 Help & Docs Forums Slack File an Issue' and language options 'English Download CLI'.

- Then click **Show advanced options** to configure **Group Wait Time** and **Group Interval Time**.
- Note: Group Wait Time is the duration of the interval to buffer alerts from the same group.
- Note: Group Interval Time is the duration taken to send the first alert to the notifier since the time of addition of the group.
- After configuring, click **Save**

This screenshot shows the same interface as the previous one, but with the 'Show advanced options' link expanded. It displays two input fields for time intervals:

- Group Wait Time:** Set to 0 minutes, 0 seconds. A note below says: "How long to wait to buffer alerts of the same group before sending initially."
- Group Interval Time:** Set to 0 minutes, 30 seconds. A note below says: "How long to wait before sending an alert that has been added to a group which contains already fired alerts."

Below these, a 'Repeat Interval Time' field is set to 1 hour, 0 minutes, 0 seconds. A note below says: "How long to wait before re-sending a given alert that has already been sent."

At the bottom right are 'Save' and 'Cancel' buttons, and a 'Hide advanced options' link.

The footer links and language options are identical to the previous screenshot.

- We have now successfully setup the Slack notifier.

## 15. Now let's setup CD pipeline with Harness using <https://harness.io>

- First, we need to setup Harness delegate. Now let's see why we need Harness delegate: The Harness Delegate is a service you run in your deployment target environment, such as your local network, VPC, or cluster. The Delegate connects all of your artifact, infrastructure, collaboration, verification and other providers with the Harness Manager. Most importantly, the Delegate performs all deployment operations.

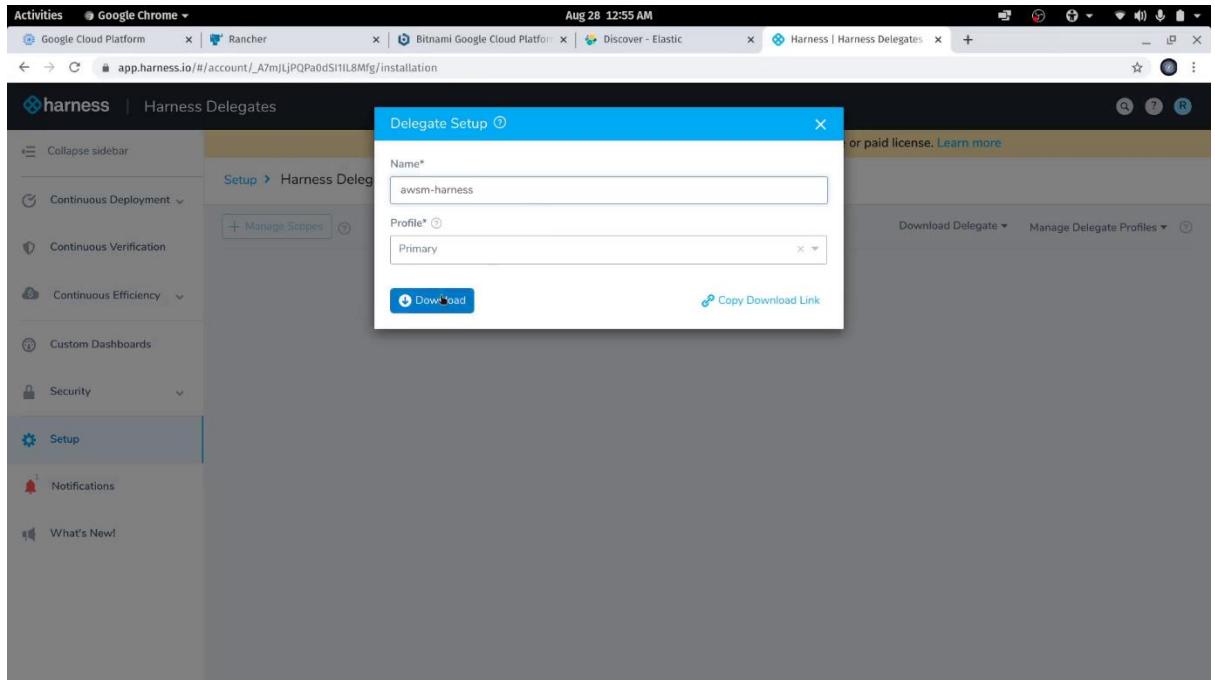
- After signing in, go to **Setup > Harness Delegates**

The screenshot shows the Harness Setup Account interface. The sidebar on the left is titled "Setup" and includes links for Continuous Deployment, Continuous Verification, Continuous Efficiency, Custom Dashboards, Security, Notifications, and What's New. The main content area has a yellow header bar stating "Your Harness Professional trial will expire in 12 days. Keep going with a free or paid license. [Learn more](#)". Below this is a "Setup" section with a "Manage Scopes" button. To the right is a sidebar titled "Account" containing links for Tags Management, Alert Notification Rules, Harness Delegates (which is highlighted), and Harness API Explorer. At the bottom of the main content area is a URL bar with the address [https://app.harness.io/#/account/\\_A7mjLjPQP0dS11L8Mfg/installation](https://app.harness.io/#/account/_A7mjLjPQP0dS11L8Mfg/installation).

- Download delegate as Docker Image but for actual use case download as Kubernetes YAML or Helm Values YAML and run the delegate in a Kubernetes environment.

The screenshot shows the Harness Harness Delegates page. The sidebar on the left is identical to the previous screenshot. The main content area has a yellow header bar with the same trial expiration message. Below it is a "Setup > Harness Delegates" section with a "Manage Scopes" button. To the right is a sidebar with "Download Delegate" and "Manage Delegate Profiles" dropdowns. A dropdown menu is open under "Download Delegate" showing options: Shell Script, Docker Image (which is highlighted), Kubernetes YAML, Helm Values YAML, and ECS Task Spec.

- Set profile as Primary as of now and download, we can change it later once the delegate is setup.



- After downloading the delegate setup, we need to extract it and open the shell script named launch-harness-delegate.sh in a text editor and change the `--hostname` value to the name of the VM instance.

```

#!/bin/bash -e
sudo docker pull harness/delegate:latest
sudo docker run -d --restart unless-stopped --hostname=awsm-harness
(echo "ACCOUNT_ID=A7mjLjPQP0dS1IIL8Mfg"; echo "ACCOUNT_SECRET=05e416f08fe078bd3977714568bf94d"; echo "MANAGER_HOST_AND_PORT=https://app.harness.io/gratis"; echo "WATCHER_STORAGE_URL=https://app.harness.io/public/free/freemium/watchers"; echo "WATCHER_CHECK_LOCATION=current.version"; echo "REMOTE_WATCHER_URL_CDON=https://app.harness.io/public/shared/watchers/builds"; echo "DELEGATE_STORAGE_URL=https://app.harness.io"; echo "DELEGATE_CHECK_LOCATION=delegatefree.txt"; echo "DELEGATE_NAME=awsm-harness"; echo "DELEGATE_PROFILE=0a6B25D0TNyBGIOc9QKrYQ"; echo "DELEGATE_TYPE=DOCKER"; echo "DEPLOY_MODE=KUBERNETES"; echo "PROXY_HOST="; echo "PROXY_PORT="; echo "PROXY_SCHEME="; echo "PROXY_USER="; echo "PROXY_PASSWORD="; echo "NO_PROXY="; echo "PROXY_MANAGER=true"; echo "POLL_FOR_TASKS=false"; echo "HELM_DESIRED_VERSION="; echo "CF_PLUGIN_HOME="; echo "USE_CDN=true"; echo "CDN_URL=https://app.harness.io"; echo "JRE_VERSION=1.8_0_242") | sed -e 's/\$/\\$/' -e 's/"/\\\"/g' -e 's/\'\\\'/\'\\\'\\\'/g' -e 's/\'\\\'\\\'/\'\\\'\\\'\\\'/g'

```

The screenshot shows a terminal window with a shell script named 'launch-harness-delegate.sh'. The script contains several environment variable assignments for a Harness delegate setup. The variables include ACCOUNT\_ID, ACCOUNT\_SECRET, MANAGER\_HOST\_AND\_PORT, WATCHER\_STORAGE\_URL, WATCHER\_CHECK\_LOCATION, REMOTE\_WATCHER\_URL\_CDON, DELEGATE\_STORAGE\_URL, DELEGATE\_CHECK\_LOCATION, DELEGATE\_NAME, DELEGATE\_PROFILE, DELEGATE\_TYPE, DEPLOY\_MODE, PROXY\_HOST, PROXY\_PORT, PROXY\_SCHEME, PROXY\_USER, PROXY\_PASSWORD, NO\_PROXY, PROXY\_MANAGER, POLL\_FOR\_TASKS, HELM\_DESIRED\_VERSION, CF\_PLUGIN\_HOME, USE\_CDN, and CDN\_URL. The script uses sed to escape double quotes and backslashes.

- Then, SSH into the VM instance created for Harness delegate. We already finished installing Docker container engine in this instance.
- Now, copy the contents of the shell script launch-harness-delegate.
- In the VM, run `sudo nano launch-harness-delegate.sh`

- Paste the copied contents
  - Ctrl+O
  - Ctrl+X
  - Now, run `sudo chmod +x Launch-harness-delegate.sh`
  - Then execute the script by running `./Launch-harness-delegate.sh`
  - Now, let's verify whether our delegate is running by using `docker ps` command.

```
Activities Terminal Aug 28 1:01AM
Open README.txt launch-harness-delegate.sh
Instructions.txt Save - + x
launch-harness-delegate.sh
-/Downloads/harness-delegate-docker
Aug 27 19:29:08 awsm-harness-delegate dockerd[3232]: time="2020-08-27T19:29:08" level=info msg="Starting up"
Aug 27 19:29:08 awsm-harness-delegate dockerd[3232]: time="2020-08-27T19:29:08" level=info msg="Docker daemon is running on unix socket"
Aug 27 19:29:08 awsm-harness-delegate dockerd[3232]: time="2020-08-27T19:29:08" level=info msg="parsed command: <none>:exec [<none>]"
Aug 27 19:29:08 awsm-harness-delegate systemd[1]: Started Docker Application Container
Aug 27 19:29:08 awsm-harness-delegate dockerd[3232]: time="2020-08-27T19:29:08" level=info msg="Lines 1-19/19 [END]"
ramprasathasokan@awsm-harness-delegate:~$ sudo usermod -aG docker ${USER}
ramprasathasokan@awsm-harness-delegate:~$ sudo su - ${USER}
ramprasathasokan@awsm-harness-delegate:~$ id -nG
ramprasathasokan@awsm-harness-delegate:~$ docker run -it alpine /bin/sh
ramprasathasokan@awsm-harness-delegate:~$ ls
ramprasathasokan@awsm-harness-delegate:~$ cd /root/.cache/harness/delegate
ramprasathasokan@awsm-harness-delegate:~$ ls
ramprasathasokan@awsm-harness-delegate:~$ ./launch-harness-delegate.sh
latest: Pulling from harness/delegate
7595c8ca8162: Pull complete
d13af8c8a98f: Pull complete
70799171dbda: Pull complete
b6c12202c5ef: Pull complete
4ef1fac6720c43: Pull complete
0b5a47720c43: Pull complete
d9900bb5f999: Pull complete
e723c5c5a75: Pull complete
72fffc1455c7: Pull complete
9928fb0a63e2: Pull complete
0638ab82fa21f: Pull complete
5442cd1fe6c: Pull complete
Digest: sha256:22c8548b30a45b1612e247f08a71c4b99d0c70362c868309e37f875f851dd96
Status: Downloaded newer image for harness/delegate:latest
docker.io/harness/delegate:latest
5f27fe99d227e10aa0f6265abb16d012d3daf15b05f8cafafffc687bd7
ramprasathasokan@awsm-harness-delegate:~$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED          STATUS              PORTS               NAMES
5f27fe99d227        harness/delegate:latest   "/bin/sh -c './start'"   22 seconds ago   Up 17 seconds          unruﬄed_mirzakhani
ramprasathasokan@awsm-harness-delegate:~$
```

- A container with image name harness/delegate:latest is currently running, so delegate setup is successful.
  - Next, let's verify the same from Harness

The screenshot shows the Harness Delegates setup page. The left sidebar has 'Setup' selected. The main area shows a summary message: 'Your Harness Professional trial will expire in 12 days. Keep going with a free or paid license.' Below this, the 'Harness Delegates' section is shown with a 'Manage Scopes' button. The delegate details are as follows:

Name	Value
Name	awsm-harness
Hostname	awsm-harness-delegate
Description	(empty)
Delegate Type	DOCKER
Profile	<b>Primary</b> (radio button selected) Startup Script Last executed at 08/28/2020 01:01 am <a href="#">View Logs</a>
IP	172.17.0.2
Status	Connected
Last heartbeat	a few seconds ago - 08/28/2020 01:01 am
Active Versions	0.1.57604
Scope Limited To	<a href="#">+ Add Scope</a>
Scope Excluded	<a href="#">+ Exclude Scope</a>
Implicit Selectors	<a href="#">awsm-harness</a> <a href="#">awsm-harness-delegate</a> <a href="#">primary</a>
Selectors	<a href="#">Edit</a>

- Now, we are going to change the delegate profile.
- Manage Delegate Profile > Add New Profile**
- Enter the following shell commands in Startup Script in order install Helm version 3.
- `curl -fsSL -o get_helm.sh`  
`https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3`
- `chmod 700 get_helm.sh`
- `./get_helm.sh`
- Then, **Submit**.

The screenshot shows the Harness Delegates setup interface. A modal window titled "Manage Delegate Profile" is open. It has fields for "Display Name" (set to "Helm V3 Install"), "Description" (set to "Helm V3 Install"), and a "Startup Script" block containing the following shell script:

```
# Install Helm V3
curl -fsSL -o get_helm.sh
https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3
chmod 700 get_helm.sh
./get_helm.sh
```

At the bottom of the modal is a "Submit" button. Below the modal, there's a list of scopes: "awsm-harness", "awsm-harness-delegate", and "primary".

- Now, change delegate profile from Primary to Helm V3 Install

The screenshot shows the Harness Delegates setup interface after the update. The delegate profile "Helm V3 Install" is now listed under the "Primary" scope. A success message at the top of the screen reads "Delegate Profile Helm V3 Install updated successfully".

- Confirm profile change.

The screenshot shows a browser window with several tabs open, including Google Cloud Platform, Rancher, Bitnami Google Cloud Platform, Home - Elastic, and Harness | Harness Delegates. The main content area displays the 'Change Delegate Profile' dialog for the 'awsmt-harness-delegate'. The dialog contains two numbered steps: 1. It may take a few minutes to apply the new profile. 2. Any installed binaries as part of the earlier profile will not be removed automatically. If you need to remove it then restart the pod for a delegate hosted in a cluster or manually clean up for a delegate hosted on a VM. Below the steps are 'Cancel' and 'Confirm' buttons. To the right of the dialog, there is a summary of the delegate's status: DOCKER, Primary IP 172.17.0.2, Connected, Last heartbeat 2 minutes ago - 08/28/2020 01:01 am, Active Versions 1.0.5704, and a list of scopes: awsmt-harness, awsmt-harness-delegate, primary. The left sidebar shows the navigation menu with 'Setup' selected.

- Now, go to **Setup > Cloud Providers**

The screenshot shows the 'Setup Account' page in the Harness interface. The left sidebar has 'Setup' selected. The main content area is titled 'Setup' and shows the 'Cloud Providers' section. It includes a search bar for 'Search Application', a 'No Results' message, and a 'Configuration As Code' button. To the right, there are sections for 'Shared Resources' (Cloud Providers, Connectors, Template Library, Application Stacks), 'Account' (Tags Management, Alert Notification Rules, Harness Delegates), and a URL at the bottom: [https://app.harness.io/#/account/\\_A7mjLjPQP0dS11L8Mfg/cloud-providers](https://app.harness.io/#/account/_A7mjLjPQP0dS11L8Mfg/cloud-providers).

- Add Cloud Provider > Kubernetes Cluster

- Go to Rancher > Cluster > awsm-cluster > Kubeconfig file

- Copy Master URL form Kubeconfig file

```

apiVersion: v1
kind: Config
clusters:
- name: "awsm-cluster"
  cluster:
    server: https://35.240.188.90/k8s/clusters/c-zmbfm
    certificate-authority-data: "LS0tLS1CRUJTIzDRVJUSUZ3Q0UFURS0tLS0tCk1JSU3pVENDQ...
      VMZ29F3SUJB20lCQURB50JnZ3foa2pPUFFREFqQtDNUnd3R2dZRFZRUJtFeE5rZVc1aGJxBoKY...
      kdsenRHnVwAh0jNkbk1sc3dHUVLVEVFRE45atLVv0YldsanJHbHpKR1Z1WlJ1dFkyRxdta...
      GNTwPbDwPREkzTVRr0d16TTxAGNTxpbBd09ESTFNVt3TxNPWVdqTdNUnd3R2dZRFZRUJtFe...
      E5rZVc1aGJxBoK1R2x6CmRHnVwAh0jNkbk1sc3dHUVLVEVFRE45atLVv0YldsanJHbHpKR...
      1Z1WlJ1dFkyRxdFKEFU0jdJcnhrakBKFJQKJnZ3fCoazpPUFFN0KJ3TNbQ0W0M95KpG6y99z...
      1EyUHf2YlRETEUEyM5P0HFK2RLyS9102tPVL1PygjY)NaVpj5v1LYzcmaJ4V2cdz3B2Rl...
      m9ee1dUUvxdzJHQ2RckSPcW57xdJVEFP0m0dvhkROEBZJfHCk3BTUNbcV3Rh2ZRFZSMFRBU...
      UgyOKFVd0f3RUIvekFLmdnchra9QUVFEOh05kFEQkdBaUvBdw0Rc3Ryak0k02FENz1RwlwT...
      ThCbWQcFJCNE1P25tjMG00VE00x0M0VDFSVFDnhMUXZXQ1FnU3J3wZvdUtyamdtc2l3VgpRY...
      3RwaiBKZGpxSGxiYkZnEWc9P0gtLs0tLlIVORCB0RVJUSUZ3Q0UFURS0tLS0t"
  users:
- name: "awsm-cluster"
  user:
    token: "kubeconfig-user-tctzr:mcbwcc7k6ckd2bwnppnqncfgbksf7rck2qf5j6fmjxxm5f8t6c97v"

contexts:
- name: "awsm-cluster"
  context:
    user: "awsm-cluster"
    cluster: "awsm-cluster"

```

- Paste the Master URL and for Authentication select Service Account Token.

Setup > Cloud Providers

Type: Kubernetes Cluster

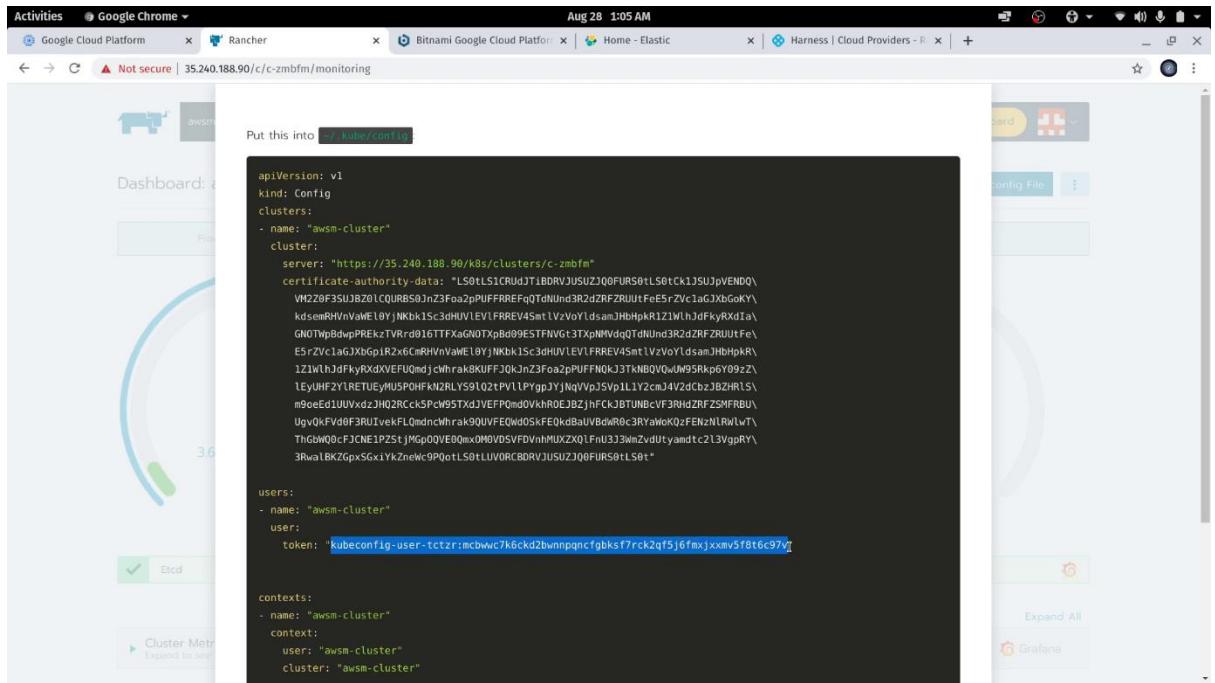
Authentication\*

- Select
- Username and password
- Service Account Token**
- OIDC Token
- Custom

All Applications	Production Environments
All Applications	Non-Production Environments

Test      Submit

- Copy the Service Account Token from Kubeconfig file

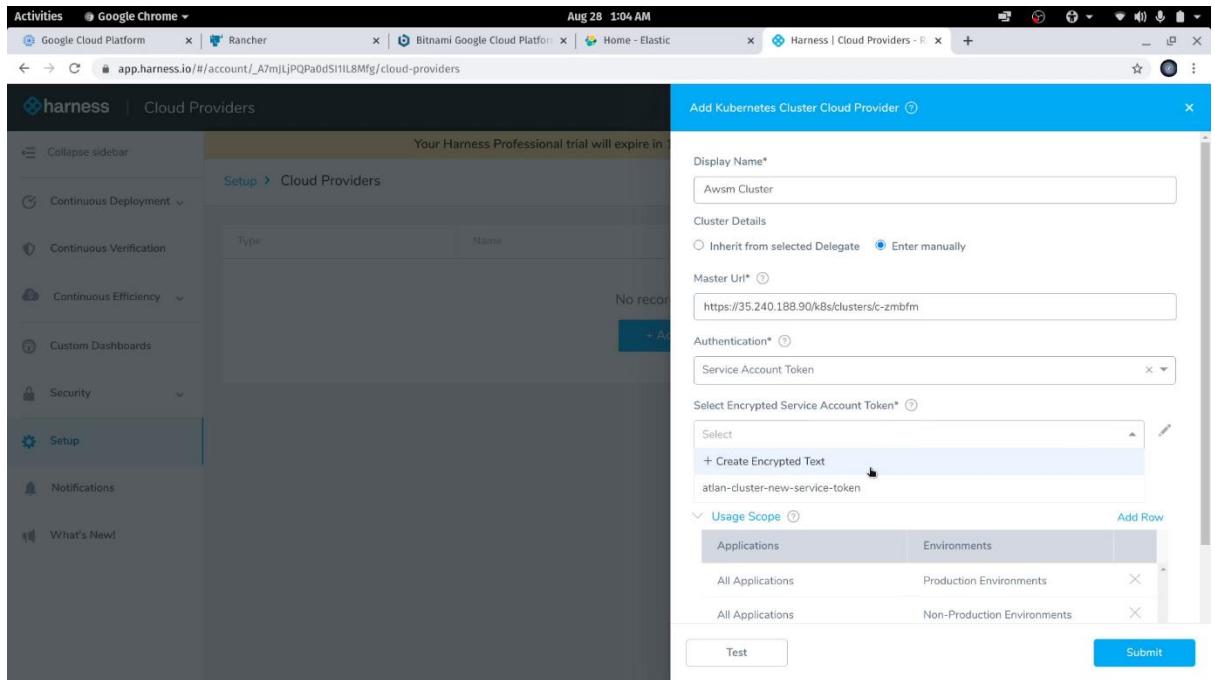


```

apiVersion: v1
kind: Config
clusters:
- name: "awsm-cluster"
  cluster:
    server: "https://35.240.188.90/k8s/clusters/c-zmbfm"
    certificate-authority-data: "LS0tLS1CRUJTIzDRVJSUZ3Q0FURS0tLS0tCk1JSU3pVENDQ...
      VMZ20F3SUJBZ0lCQURBS0JnZ3Foa2pPUFFREFqTqdNuHd3R2dZRFZRUJtFeE5rZVc1aGJxGoKY...
      kdsenRH/nvAwI0YjNKbk1sc3dHUVLEVFRREV45atLzvYoYlsanJHbHpKR1Z1WlJ1dFkyRXdta...
      GNTwPbDwPREkzTVRd016TTxgN0NTxpBd09ESTFNVgt3TxnWVdqTdNuHd3R2dZRFZRUJtFe...
      E5rZVc1aGJxGpIR2x6CmRHnVawE10YjNKbk1sc3dHUVLEVFRREV45atLzvYoYlsanJHbHpKR...
      1Z1WlJ1dFkyRXdFVEFU0mJcWnraK0UFJQkJnZ3Foa2pPUFFN0k3JTNbQ0w0M95Rkp6y09z...
      1EyUhF2YlRETEUEyU5P0HFK2RLyS9l02tPVL1PygpJY)NaVp35v1LYzcJ4V2cdzB2HLr...
      m9ee1dUUvxdzJHQ2RckSPcW5TxdJVEFP0m0dvhkROEBZJfFcK3BTUNbcV3RhHzFRBvU...
      UgyOKFVd0f3RUIveFL0mdnchra9QUVFEOw0s05kFEQkdBaUvBdw0c0c3Ryak0k02FENz1Rw...
      ThObmQ0cFJCNc1P25tjMG00VE00mx0M0VSDVFdnhMuxZXQ1FnU3J3wZvdUyamdtc2l3Vg...
      3RwaiBKZ6pxSGxiYkZnEWc9P0gtLlIVORCB0RJUSUZ3Q0FURS0tLS0t"
users:
- name: "awsm-cluster"
  user:
    token: "kubeconfig-user-tctzr:mcbwvc7k6ckd2bwnppnqncgbksf7rck2qf5j6fmjxxm5f8t6c97vA"
contexts:
- name: "awsm-cluster"
  context:
    user: "awsm-cluster"
    cluster: "awsm-cluster"

```

- Now, let's see how to Create Encrypted Text in Harness and use it



Add Kubernetes Cluster Cloud Provider

Display Name\*  
Awsm Cluster

Cluster Details  
 Inherit from selected Delegate  Enter manually

Master Url\*  
https://35.240.188.90/k8s/clusters/c-zmbfm

Authentication\*  
Service Account Token

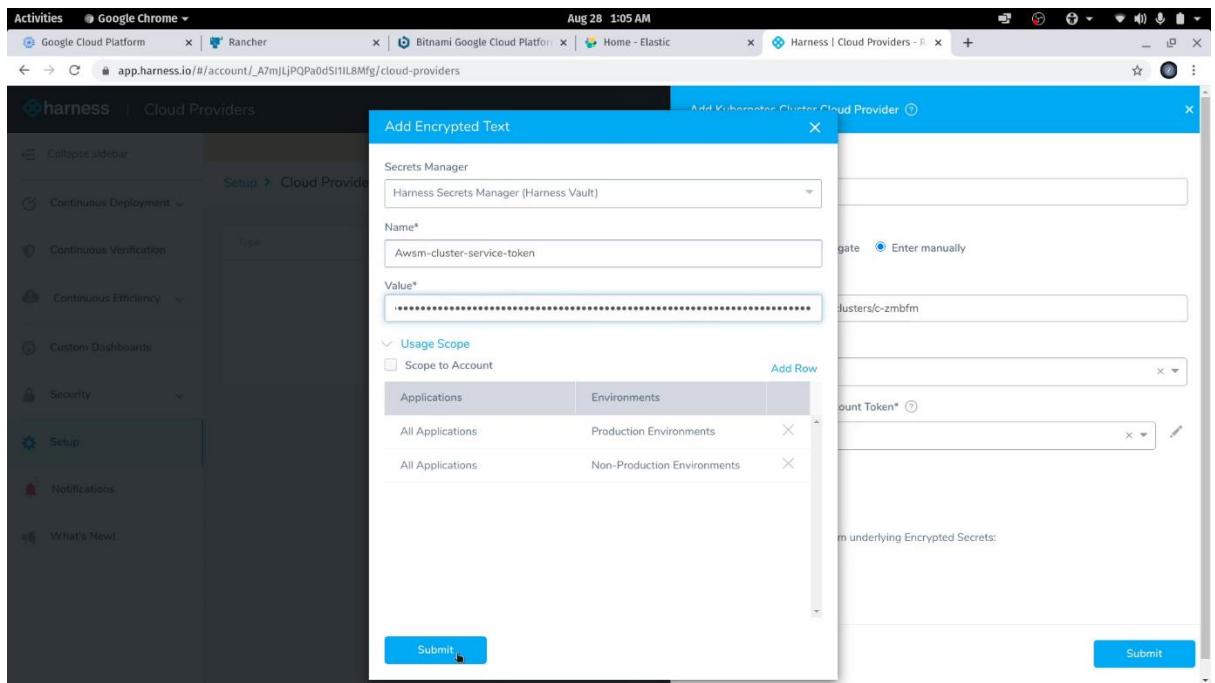
Select Encrypted Service Account Token\*  
+ Create Encrypted Text  
atlan-cluster-new-service-token

Usage Scope  

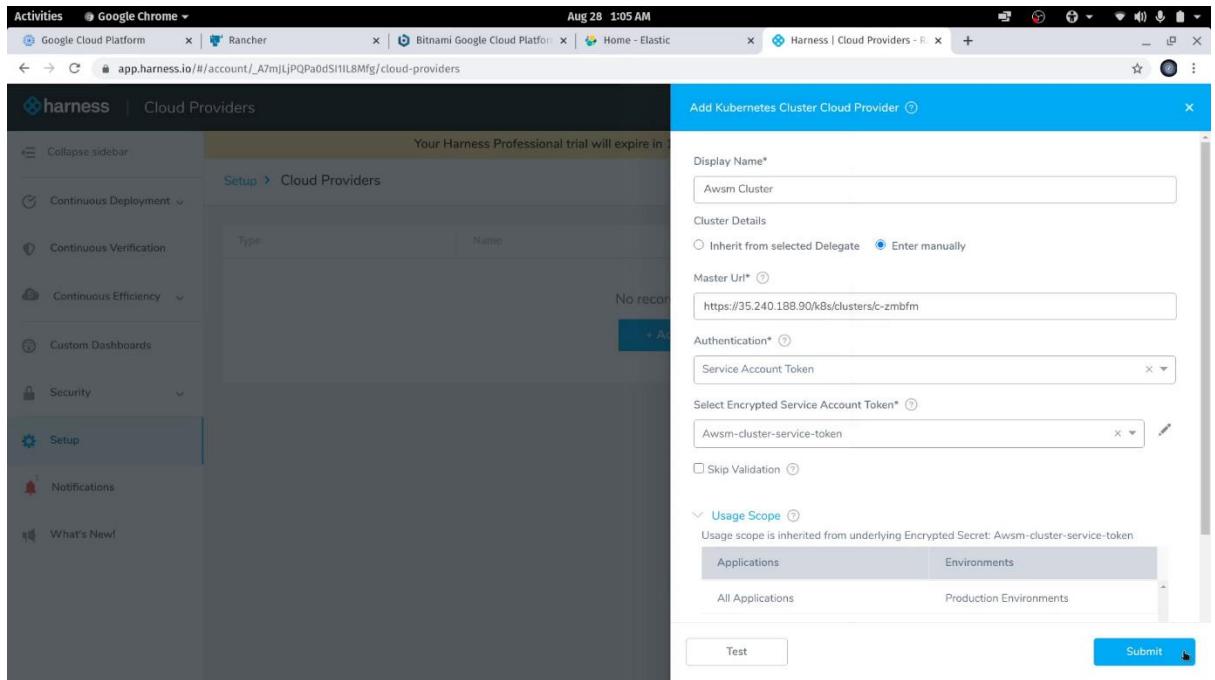
Applications	Environments
All Applications	Production Environments
All Applications	Non-Production Environments

**Submit**

- Paste the copied **Service Account Token** value in the **Value** field and **Submit**.



- After creation of the encrypted text, select it in the **Encrypted Service Account Token** field and **Submit**.



- Now, our Kubernetes cluster is successfully added.

The screenshot shows the Harness Cloud Providers interface. On the left, a sidebar menu includes options like Activities, Continuous Deployment, Continuous Verification, Continuous Efficiency, Custom Dashboards, Security, Setup (which is selected), Notifications, and What's New!. The main content area displays a table titled "Cloud Providers". A single row is present, representing an AWS Lambda cluster. The columns show the Type (Kubernetes), Name (Aws Lambda Cluster), and URL (https://35.240.188.90/k8s/clusters/c-zmbfm). A "Details" button is also visible. At the top of the main area, a message states: "Your Harness Professional trial will expire in 12 days. Keep going with a free or paid license. Learn more".

- Next, let's go to **Settings > Connectors > Artifact Servers** add our artifact server from where we are going to retrieve the Helm Charts for deploying the parse server.

The screenshot shows the Harness Setup Account interface. The sidebar is identical to the previous one. The main content area has a header "Setup" with a "Configuration As Code" link. Below it is a section titled "Applications" with a search bar and a "+ Add Application" button. To the right, there is a "Shared Resources" sidebar containing links for Cloud Providers, Connectors (which is selected and highlighted in blue), Template Library, and Application Stacks. Further down the page, under the "Account" section, are links for Tags Management, Alert Notification Rules, and Harness Delegates. At the bottom of the page, the URL https://app.harness.io/#/account/\_A7mjLjPQP0dS11L8Mfg/connectors is displayed.

- Now let's add our artifact server by click **Add Artifact Server**.

The screenshot shows the Harness interface for managing artifact servers. On the left, there's a sidebar with various options like Continuous Deployment, Continuous Verification, Continuous Efficiency, Custom Dashboards, Security, and Setup. The Setup option is currently selected. In the main content area, under 'Setup > Connectors > Artifact Servers', a table lists an existing artifact server: 'Type' is Docker, 'Name' is 'Harness Docker Hub', and 'URL' is 'https://registry.hub.docker.com/v2/'. There's also a '+ Add Artifact Server' button.

- Let's add our Helm repository details
- The URL of Bitnami Helm repository is <https://charts.bitnami.com/bitnami>
- No Username/Password/Token is necessary
- Click **Submit**

The screenshot shows the 'Helm Repository' configuration dialog. It has several fields: 'Type' set to 'Helm Repository', 'Display Name' set to 'Bitnami Helm Hub', 'Hosting Platform' set to 'HTTP Server', and 'Repository URL' set to 'https://charts.bitnami.com/bitnami'. Below these, there's a 'Username' field which is empty. Under 'Select Encrypted Password/Token', there's a dropdown menu set to 'Select'. At the bottom, there's a 'Usage Scope' section with two tabs: 'Applications' and 'Environments'. The 'Applications' tab is selected. A 'Submit' button is located at the bottom right.

- We have successfully added our Bitnami Helm Repository in our Artifact Server.

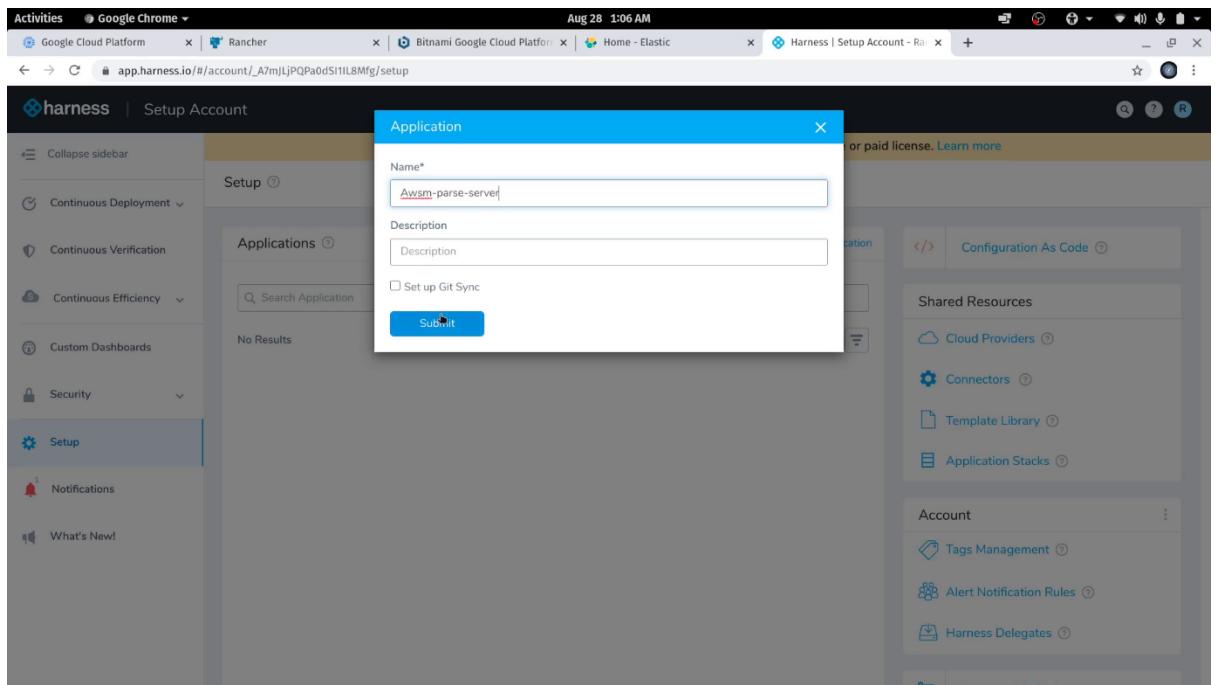
The screenshot shows the Harness web interface. The left sidebar is collapsed. The main navigation bar at the top includes tabs for Google Cloud Platform, Rancher, Bitnami Google Cloud Platform, Home - Elastic, and Harness | Artifact Servers. The current page is 'Artifact Servers' under 'Setup > Connectors'. A message at the top right says 'Your Harness Professional trial will expire in 12 days. Keep going with a free or paid license.' Below this, there's a table listing artifact servers:

Type	Name	URL	Action
Docker	Harness Docker Hub	<a href="https://registry.hub.docker.com/v2/">https://registry.hub.docker.com/v2/</a>	⋮
Helm Repository	Bitnami Helm Hub	<a href="https://charts.bitnami.com/bitnami">https://charts.bitnami.com/bitnami</a>	⋮

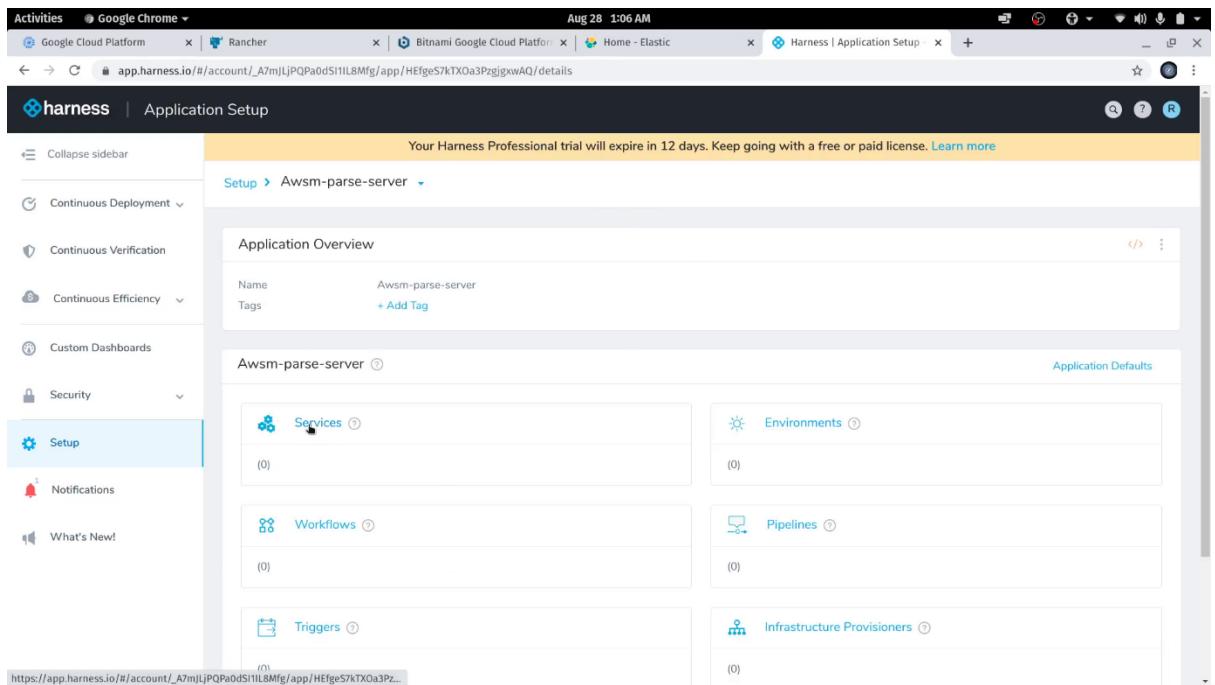
- Now, let's Add Application from Setup.

The screenshot shows the Harness 'Setup Account' page. The left sidebar is collapsed. The main navigation bar at the top includes tabs for Google Cloud Platform, Rancher, Bitnami Google Cloud Platform, Home - Elastic, and Harness | Setup Account. The current page is 'Setup'. On the left, there's a 'Setup' section with a 'Search Application' input field and a 'No Results' message. To the right, there are sections for 'Configuration As Code' (with a 'Create New' button), 'Shared Resources' (Cloud Providers, Connectors, Template Library, Application Stacks), and 'Account' (Tags Management, Alert Notification Rules, Harness Delegates).

- Enter Application Name and Submit.



- Next, go to **Awsom-parse-server > Services**



- Now, let's Add Service.

- Add a name for our service.
- Choose **Kubernetes** under **Deployment Type** since we are deploying in a Kubernetes Cluster.
- Then, **Submit**.

- Let us link our Bitnami Parse Server Manifest

The screenshot shows the Harness Setup Services interface. On the left is a sidebar with various options like Continuous Deployment, Continuous Verification, and Security. The main area shows a manifest tree under the 'deployment' folder. A context menu is open over the 'deployment.yaml' file, with options: 'Link Remote Manifests', 'Upload Inline Manifest Files', and 'Delete All Manifest Files'. The code for deployment.yaml is partially visible:

```

1 {{- if .Values.env.config}}
2   apiVersion: v1
3   kind: ConfigMap
4   metadata:
5     name: {{.Values.name}}
6   data:
7     {{.Values.env.config | toYaml | indent 2}}
8   ...
9 {{- end}}
10 {{- if .Values.env.secrets}}
11   apiVersion: v1
12   kind: Secret
13   metadata:
14     name: {{.Values.name}}
15   stringData:
16     {{.Values.env.secrets | toYaml | indent 2}}
17   ...
18 {{- end}}
19 {{- if .Values.dockercfg}}
20
21 {{- if .Values.dockercfg}}

```

- Add the chart specifications in the Remote Manifests form.
- Chart Name: bitnami/parse
- Chart Version: 11.0.4
- Then, **Submit**.

The screenshot shows the 'Remote Manifests' dialog box overlaid on the main Harness interface. The dialog has the following fields:
 

- Manifest Format: Helm Chart from Helm Repository
- Helm Repository: Bitnami Helm Hub (<https://charts.bitnami.com/bitnami>)
- Chart Name: bitnami/parse
- Chart Version: 11.0.4
- Helm Version: v3

 At the bottom of the dialog is a 'Submit' button. The background of the main interface shows the same manifest tree and code snippet as the previous screenshot.

- Our manifest is successfully updated.

The screenshot shows the Harness Setup Services interface. The left sidebar includes options like Continuous Deployment, Continuous Verification, Continuous Efficiency, Custom Dashboards, Security, and Setup (which is selected). The main panel displays setup details for a service named "bitnami-parse-server". It shows the Manifest Format as Helm Chart from Helm Repository, Helm Repository as Bitnami Helm Hub, Chart Name as bitnami/parses, and Chart Version as 11.0.4. Below this, there are sections for Configuration, including Config Variables, Config Files, and Values YAML Override.

- Now, let's head to Environments.

The screenshot shows the Harness Application Setup interface. The left sidebar has the "Setup" option selected. The main panel shows an "Application Overview" for the application "Awm-parseserver". It lists the name and tags. Below this, under the heading "Awm-parseserver", there are several sections: Services (with one entry: bitnami-parse-server), Environments (empty), Workflows (empty), Pipelines (empty), Triggers (empty), and Infrastructure Provisioners (empty).

- Next, we will Add Environment.

The screenshot shows the Harness interface with the sidebar expanded. The 'Setup' option is selected. The main content area displays a message: 'Your Harness Professional trial will expire in 12 days. Keep going with a free or paid license. [Learn more](#)'. Below this, the breadcrumb navigation shows 'Setup > Awsmp-parse-server > Environments'. A search bar and a '+ Add Environment' button are visible. The central message 'No Results' is followed by 'There are no Environments.' and a prominent blue 'Add Environment' button.

- Select **Environment Type** as **Production** and name the Environment.
- Then, **Submit**.

The screenshot shows the Harness interface with the sidebar expanded. The 'Setup' option is selected. A modal dialog box titled 'Environment' is open in the center. It contains fields for 'Name\*' (with 'awsmp-parse' entered), 'Description' (empty), and 'Environment Type\*' (set to 'Production'). A 'Submit' button is at the bottom of the dialog. The background shows the same Harness setup environment page as the previous screenshot.

- Now, let's Add Infrastructure Definition.

The screenshot shows the Harness Setup Environments interface. On the left, there is a sidebar with various navigation options: Activities, Continuous Deployment, Continuous Verification, Continuous Efficiency, Custom Dashboards, Security, Setup, Notifications, and What's New. The main content area is titled "Environment Overview" and shows details for an environment named "awsm-parse". It indicates the environment type is "Production" and has no tags. Below this is the "Infrastructure Definitions" section, which currently displays "No Infrastructure Definition." A prominent blue button labeled "+ Add Infrastructure Definition" is centered in this section. Further down, there is a "Service Configuration Overrides" section with a similar "No Service Configuration Overrides" message and a "+ Add Configuration Overrides" button.

- Fill in the following details in the **Infrastructure Definition** form and **Submit**.

The screenshot shows the "Infrastructure Definition" dialog box overlaid on the Harness interface. The dialog contains the following fields:

- Name\***: awsm-cluster-infra
- Cloud Provider Type\***: Kubernetes Cluster
- Deployment Type\***: Kubernetes (including Helm, OpenShift, etc.)
- Use Already Provisioned Infrastructure** (radio button selected)
- Cloud Provider\***: Kubernetes Cluster: Awsm Cluster
- Namespace**: default
- Release Name\***: release-\${infra.kubernetes.infradl}
- Scope to Specific Services** (checkbox)

A blue "Submit" button is at the bottom of the dialog.

- We have successfully added our **Infrastructure Definition**.

The screenshot shows the Harness web interface. The left sidebar is collapsed. The main navigation bar includes 'Activities', 'Google Chrome', 'Rancher', 'Bitnami Google Cloud Platform', 'Home - Elastic', and 'Harness | Setup Environment'. The current page is 'Setup Environments' under 'Continuous Deployment'. A green success message 'Added Successfully' is displayed at the top right. Below it, a yellow banner says 'Your Harness Professional trial will expire in 12 days. Keep going with a free or paid license. [Learn more](#)'. The main content area shows an 'Environment Overview' for 'awsm-parse' with details: Name: awsm-parse, Environment Type: Production, Tags: + Add Tag. Under 'Infrastructure Definitions', there is one entry: 'awsm-cluster-infra' with Deployment Type: Kubernetes (including Helm, OpenShift, etc.), Namespace: default, and Scope to Specific Services: No. The 'Service Configuration Overrides' section shows 'No Service Configuration Overrides.' and a blue 'Add Configuration Overrides' button.

- Next, **Awsm-parse-server > Workflows**

The screenshot shows the Harness 'Application Setup' page for the 'Awsm-parse-server' environment. The left sidebar is expanded, showing 'Setup' selected. The main content area has a yellow banner with the same trial expiration message. The 'Awsm-parse-server' section contains tabs for 'Services' (1 bitnami-parse-server), 'Environments' (1 awsm-parse), 'Workflows' (0), 'Pipelines' (0), 'Triggers' (0), and 'Infrastructure Provisioners' (0). Below these is an 'Application Resources' section with a 'Template Library' link.

- **Add Workflow**

The screenshot shows a browser window with the URL [https://app.harness.io/#/account/\\_A7mjLjPQP0dS1IL8Mfg/app/HFgeS7kTXOa3PzgjgxwAQ/workflows](https://app.harness.io/#/account/_A7mjLjPQP0dS1IL8Mfg/app/HFgeS7kTXOa3PzgjgxwAQ/workflows). The page title is "Setup Workflows". On the left, there is a sidebar with "Setup" selected. The main content area displays a message: "Your Harness Professional trial will expire in 12 days. Keep going with a free or paid license. [Learn more](#)". Below this, it says "No Results" and "There are no Workflows.". A blue "Add Workflow" button is visible. The top of the browser shows tabs for Google Cloud Platform, Rancher, Bitnami Google Cloud Platform, Home - Elastic, and Harness | Setup Workflows.

- Select **Workflow Type as Rolling Deployment**.
- Fill the rest of the form.
- Then, **Submit**.

The screenshot shows the same browser window with the "Add Workflow" dialog open. The dialog has a blue header "Workflow". The form fields are filled as follows:

- Name\*: awsm-parse-rolling-deployment
- Description: Enter a Description (optional)
- Workflow Type\*: Rolling Deployment
- Environment\*: awsm-parse
- Service\*: bitnami-parse-server
- Infrastructure Definition\*: awsm-cluster-infra

A blue "Submit" button is at the bottom of the dialog. The background of the browser shows the same Harness interface as the previous screenshot.

- Workflow successfully added.

The screenshot shows the Harness interface for setting up a workflow. The left sidebar is collapsed. The main navigation path is: Setup > Awsm-parse-server > Workflows > awsm-parse-rolling-deployment > Rolling. The workflow type is set to 'Rolling'. The workflow consists of three steps: 1. Deploy, 2. Verify, and 3. Wrap Up. Each step has an 'Add Step' button. To the right of the steps, there are sections for Rollback Steps, Notification Strategy, Failure Strategy, Concurrency Strategy, and Workflow Variables. A message at the top states: "Your Harness Professional trial will expire in 12 days. Keep going with a free or paid license. [Learn more](#)".

- Finally, let's head to Awsm-parse-server > Pipelines

The screenshot shows the Harness Application Setup page for the 'Awsm-parse-server' application. The sidebar is collapsed. The main view shows the 'Application Overview' section with the application name 'Awsm-parse-server' and tags '(1) awsm-parse'. Below this, there are several sections: Services (1) bitnami-parse-server, Environments (1) awsm-parse, Workflows (1) awsm-parse-rolling-deployment, Pipelines (0), Triggers (0), and Infrastructure Provisioners (0). A link at the bottom left provides the full URL: [https://app.harness.io/#/account/\\_A7mjLjPQP0dS11L8Mfg/app/HFgeS7kTXOa3Pz...](https://app.harness.io/#/account/_A7mjLjPQP0dS11L8Mfg/app/HFgeS7kTXOa3Pz...). A message at the top states: "Your Harness Professional trial will expire in 12 days. Keep going with a free or paid license. [Learn more](#)".

- Now, let's Add Pipeline.

Activities   Google Chrome ▾ Aug 28 1:10 AM

Google Cloud Platform | Rancher | Bitnami Google Cloud Platfo... | Home - Elastic | Harness | Setup Pipelines - R

app.harness.io/#/account/\_A7mjLjPQP0dS1IL8Mfg/app/HFgeS7kTXOa3PzgjgxwAQ/pipelines

**harness** | Setup Pipelines

Collapse sidebar

Continuous Deployment

Continuous Verification

Continuous Efficiency

Custom Dashboards

Security

**Setup**

Notifications

What's New!

Setup > Awsrm-parse-server > Pipelines

No Results

Your Harness Professional trial will expire in 12 days. Keep going with a free or paid license. [Learn more](#)

Search

+ Add Pipeline

Add Pipeline

There are no Pipelines.

- Enter a name for our Pipeline and Submit.

Activities   Google Chrome ▾ Aug 28 1:10 AM

Google Cloud Platform | Rancher | Bitnami Google Cloud Platfo... | Home - Elastic | Harness | Setup Pipelines - R

app.harness.io/#/account/\_A7mjLjPQP0dS1IL8Mfg/app/HFgeS7kTXOa3PzgjgxwAQ/pipelines

**harness** | Setup Pipelines

Collapse sidebar

Continuous Deployment

Continuous Verification

Continuous Efficiency

Custom Dashboards

Security

Setup

Notifications

What's New!

Setup > Awsrm-parse-server > Pipelines

No Results

Add Pipeline

Add Pipeline

Name\* Deploy-awsrm-parse

Description

Submit

or paid license. [Learn more](#)

- Next, we are going to add an **Approval Stage**.

The screenshot shows the Harness web interface. The sidebar on the left has 'Setup' selected. The main content area is titled 'Pipeline Overview'. It shows a pipeline named 'Deploy-awsm-parse' with one tag, '+ Add Tag'. Below this is a section titled 'Pipeline Stages' with a button labeled 'Click to add Pipeline Stage'.

- Harness provides us with User Group(s) for enforcing RBAC (Role Based Access Control).
- Click **Submit**.

The screenshot shows the 'Add Pipeline Stage' dialog box. The 'Approval Step' radio button is selected. The 'User Group(s)' dropdown contains 'Account Administrator'. There is a note below stating 'Please ensure that all User Groups have Application: Deployments: read permission.' The 'Submit' button at the bottom is highlighted with a cursor.

- Now let's add the last stage of our pipeline and it is the **Execution Step**.

The screenshot shows the Harness web interface for setting up pipelines. On the left, there is a sidebar with various options like Continuous Deployment, Continuous Verification, Continuous Efficiency, Custom Dashboards, Security, and Setup. The Setup option is currently selected. The main area is titled "Pipeline Overview" and shows a pipeline named "Deploy-awsm-parse". Below the pipeline name, there is a section for "Pipeline Stages" with a button labeled "Click to add Pipeline Stage". There is also a "STAGE 1" box containing an "Approval" step.

- Select our rolling deployment workflow which we created under **Execute Workflow**.
- Then, **Submit**.

The screenshot shows the Harness web interface with a modal dialog box open. The dialog is titled "Execution Step" and contains fields for "Step Name\*" (set to "awsm-parse-rolling-deployment") and "Execute Workflow\*" (set to "awsm-parse-rolling-deployment"). There are also checkboxes for "Auto Generate Name" and "Execute in Parallel with Previous Step", and a dropdown for "Option to Skip Step" set to "Do not skip". At the bottom of the dialog is a "Submit" button.

- We have successfully created our CD pipeline using Harness.

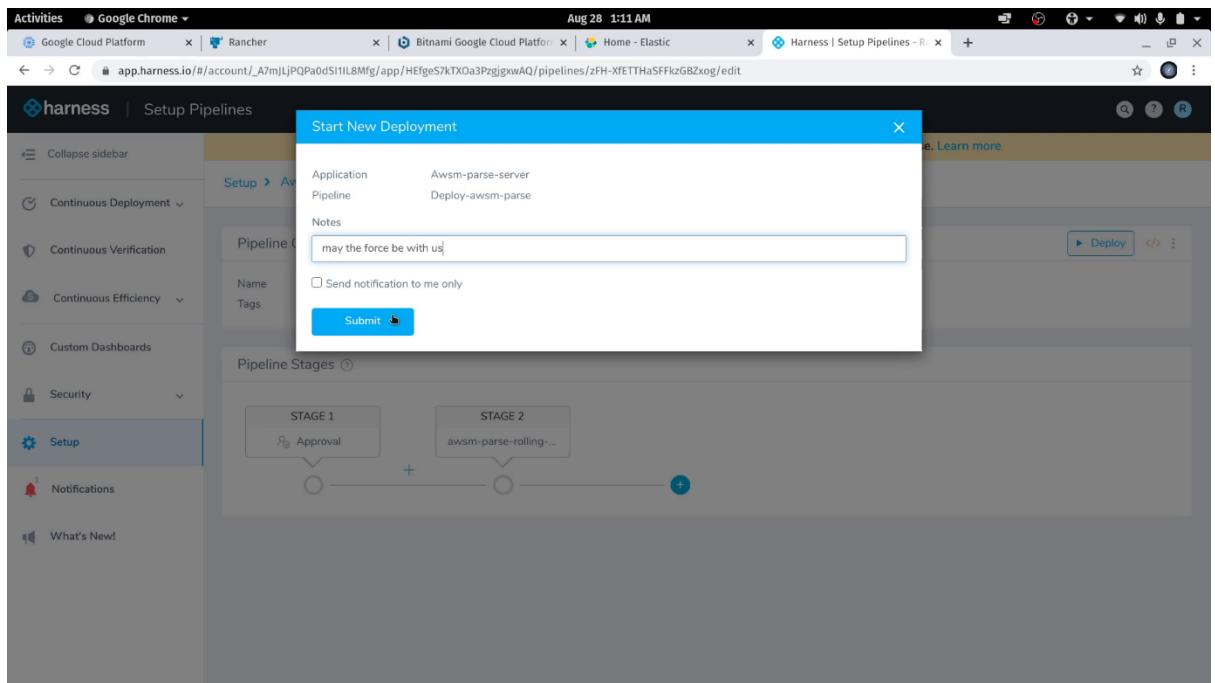
The screenshot shows the Harness web interface for setting up pipelines. On the left, there's a sidebar with various options like Continuous Deployment, Continuous Verification, and Security. The 'Setup' option is currently selected. The main area is titled 'Pipeline Overview' for a pipeline named 'Deploy-awsm-parse'. It shows two stages: 'STAGE 1' containing an 'Approval' step and 'STAGE 2' containing an 'awsms-parse-rolling...' step. A horizontal line connects the stages. At the top right of the pipeline view, there's a blue 'Deploy' button. The browser's address bar shows the URL for the Harness pipeline setup.

16. Now, let's deploy our Parse Server Application using rolling update strategy i.e. incremental update strategy.

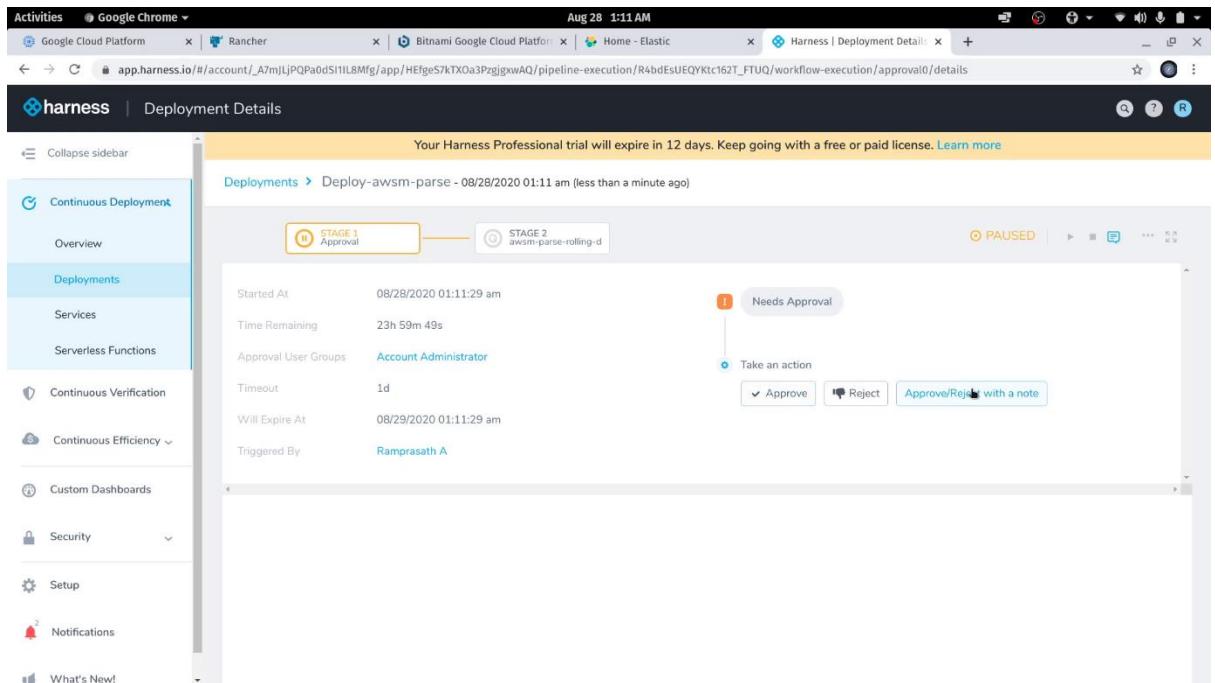
- Click Deploy.

This screenshot is nearly identical to the previous one, showing the Harness pipeline setup interface. The difference is that the 'Deploy' button at the top right of the pipeline overview is now highlighted in blue, indicating it is the active or selected action. The rest of the interface, including the pipeline stages and the sidebar, remains the same.

- Add a note if required and **Submit**.



- Let's approve with a note.
- Note: Once we trigger **Deploy**, an email will be sent by Harness notifying regarding deployment and its approval for the Deployment.



- Click Approve.

The screenshot shows the Harness Deployment Details page. On the left, a sidebar menu is open under 'Continuous Deployment' with options like Overview, Deployments (which is selected), Services, Serverless Functions, Continuous Verification, Continuous Efficiency, Custom Dashboards, Security, Setup, and Notifications. A 'What's New?' section is also present. The main content area displays a deployment titled 'Deploy-awsm-parse - 08/28/2020 01:11 am (less than a minute ago)'. The deployment status is 'PAUSED'. It shows two stages: 'STAGE 1 Approval' (highlighted in yellow) and 'STAGE 2 awsm-parse-rolling-d'. Below the stages, deployment details are listed: Started At (08/28/2020 01:11:29 am), Time Remaining (23h 59m 32s), Approval User Groups (Account Administrator), Timeout (1d), Will Expire At (08/29/2020 01:11:29 am), and Triggered By (Ramprasath A). A note says 'deployment approved.' with an 'Approve' button. A 'Needs Approval' badge is also visible.

- Deployment has been approved.

The screenshot shows the same Harness Deployment Details page as before, but the deployment status is now 'RUNNING'. The deployment details remain the same: Started At (08/28/2020 01:11:29 am), Duration (30s), Ended At (08/28/2020 01:11:59 am), Approval User Groups (Account Administrator), Timeout (1d), Triggered By (Ramprasath A), and Approved By (Ramprasath A). A note says 'deployment approved.' with an 'Approved By Ramprasath A (ramprasathasokan@gmail.com)' entry. The 'Needs Approval' badge is no longer present.

- Deployment in progress.

The screenshot shows the Harness Deployment Details page for a deployment named "Deploy-awsm-parse" initiated on Aug 28 1:11 AM. The deployment consists of two stages: STAGE 1 (Approval) and STAGE 2 (awsm-parse-rolling-c). Stage 1 is completed (Status: RUNNING). Stage 2 is in progress (Status: RUNNING, Progress: 33%). The workflow diagram illustrates the steps: Pre-Deployment → Rolling → Deploy → Acquire Resource Lock → Rollout Deployment. The Rollout Deployment step is currently active. The right panel displays deployment artifacts, infrastructure, and a log of the rollout process, which includes fetching files and applying changes.

- Deployment successful.

The screenshot shows the Harness Deployment Details page for the same deployment "Deploy-awsm-parse" now successfully completed on Aug 28 1:11 AM (3 mins ago). The deployment status is now SUCCESS. The workflow diagram shows the completed steps: Pre-Deployment → Rolling → Deploy → Acquire Resource Lock → Rollout Deployment → Verify → Wrap Up → Post-Deployment. The right panel shows the final deployment artifacts, infrastructure, and a log of the successful deployment process.

- Let's verify our deployment from Rancher > Cluster > awsm-cluster > Workloads

Rancher Workloads Overview

Pod	Image	Status	Replicas
release-9fc2754b-2b23-331e-ac86-92eed3727978-mongod...	docker.io/bitnami/mongodb:4.2.8-debian-10-r46	1 Pod / Created 2 minutes ago / Pod Restarts: 0	1
release-9fc2754b-2b23-331e-ac86-92eed3727978-parse-s...	docker.io/bitnami/parse:4.3.0-debian-10-r10 1337/tcp, 80/tcp	1 Pod / Created 2 minutes ago / Pod Restarts: 0	1

- We have successfully deployed our parse server application using rolling update strategy or incremental update strategy.
- We can rollback our deployment from Rancher if needed.

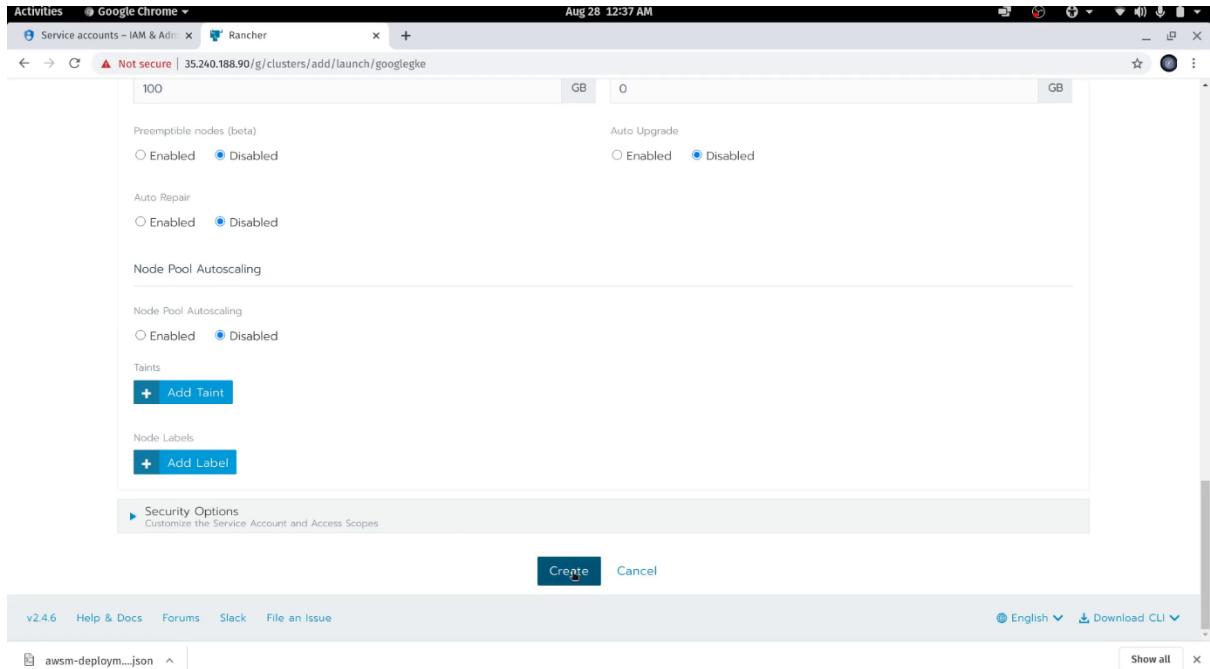
17. Note: Due to extensive time spent on learning new concepts and new tools from scratch, I was not able to setup Velero for backing up data.

## Performance and Scaling

As far as horizontal scaling is concerned, we can do it manually after provisioning the Kubernetes cluster or enable node pool autoscaling feature while adding a new cluster from GKE. I was not able to enable node pool autoscaling due to the fact that I have a maximum quota of 8vCPUs per region in GCP and I have used all of them by provisioning 4 nodes.

What node pool autoscaling feature does is increases or decreases the size of the node pool automatically, based on the resource requests (rather than actual resource utilization) of Pods running on that node pool's nodes.

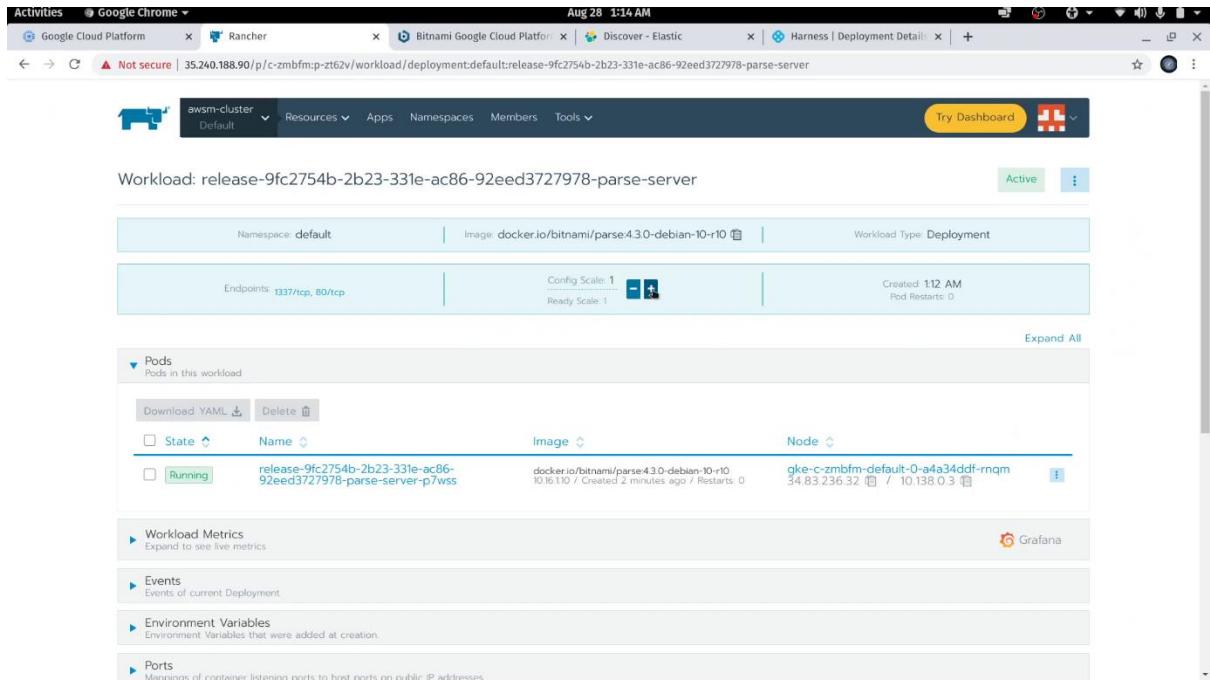
To enable node pool autoscaling, select enable under node pool autoscaling



The screenshot shows the Rancher interface for managing a node pool. In the 'Node Pool Autoscaling' section, the 'Enabled' radio button is selected. Other configuration options like 'Preemptible nodes (beta)' and 'Auto Repair' are also present.

To manually scale a deployed app, we can increase the number of instances by the following steps:

- Go to **Rancher > Cluster > awsm-cluster > default (namespace) > workloads > release-id-parse-server**



The screenshot shows the Rancher interface for a workload named 'release-id-parse-server'. It displays the deployment configuration, endpoints (1337/tcp, 80/tcp), and a single pod named 'release-9fc2754b-2b23-331e-ac86-92eed3727978-parse-server-p7wss' running on node 'gke-c-zmbfm-default-0-a4a34ddf-rmqm'.

- We can see that a new instance of a container is being created

State	Name	Image	Node
Unavailable	release-9fc2754b-2b23-331e-ac86-92eed3727978-parse-server-s7mfd	docker.io/bitnami/parse:4.3.0-debian-10-r10 Created a few seconds ago / Restarts: 0	gke-c-zmbfm-default-0-a4a34ddf-vckm 34.105.56.70 / 10.138.0.5
Running	release-9fc2754b-2b23-331e-ac86-92eed3727978-parse-server-p7wss	docker.io/bitnami/parse:4.3.0-debian-10-r10 10.16.110 / Created 2 minutes ago / Restarts: 0	gke-c-zmbfm-default-0-a4a34ddf-rnqm 34.83.236.32 / 10.138.0.3

## Unresolved Issues / Recommended Approaches

1. Running Rancher in a Kubernetes cluster using RKE for high availability and scalability.
2. Running Harness delegate in a Kubernetes cluster for high availability and scalability.
3. Running ELK Stack in a self-hosted instance in GCP instead of using Bitnami Launchpad.
4. Enable node pool autoscaling while adding a new cluster for horizontal scaling.
5. Using self-hosted version of Harness to comply with privacy guidelines and to protect IP.
6. Incorporating OWASP ModSecurity WAF rules by adding annotation in Ingress if NGINX-Ingress is used as L4 load balancer.
7. Application source code protection can be achieved by using code obfuscation.

## Other tools that I tried

1. GitLab
2. RKE (Rancher Kubernetes Engine)
3. Weave Cloud

## References

1. <https://rancher.com/docs/rancher/v2.x/en/>
2. <https://docs.harness.io/>
3. <https://www.elastic.co/guide/index.html>
4. <https://helm.sh/docs/>