

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. 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

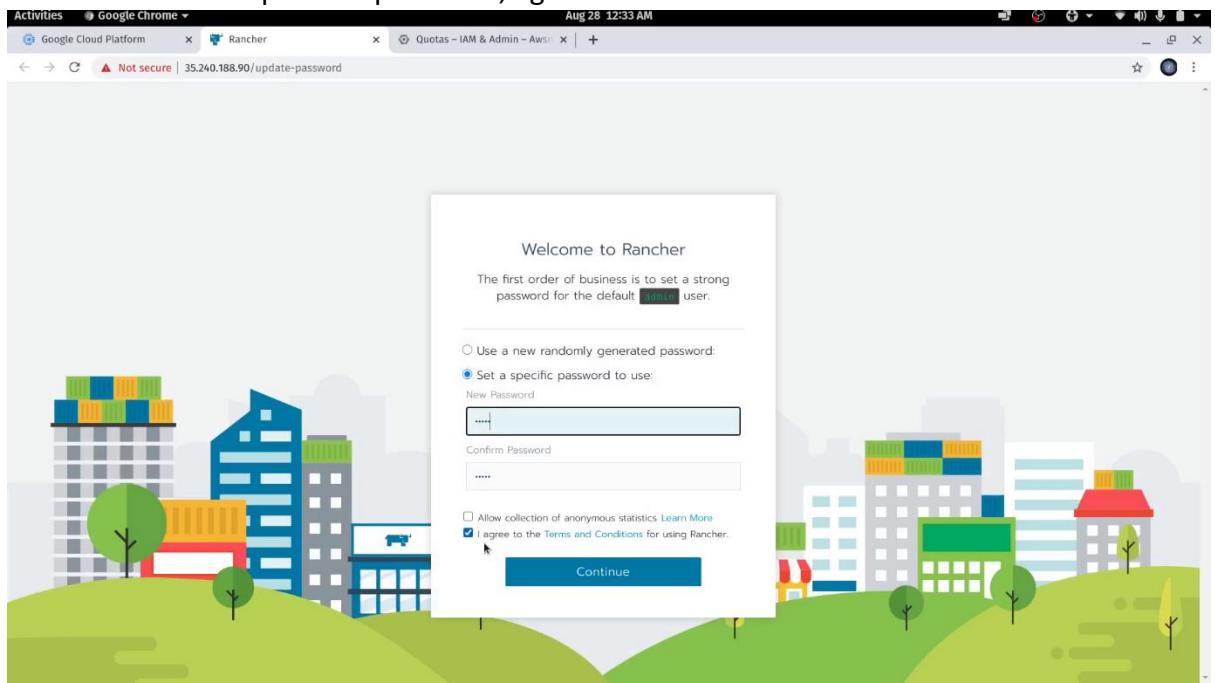
YouTube link: <https://youtu.be/FMxPRegLg-A>

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

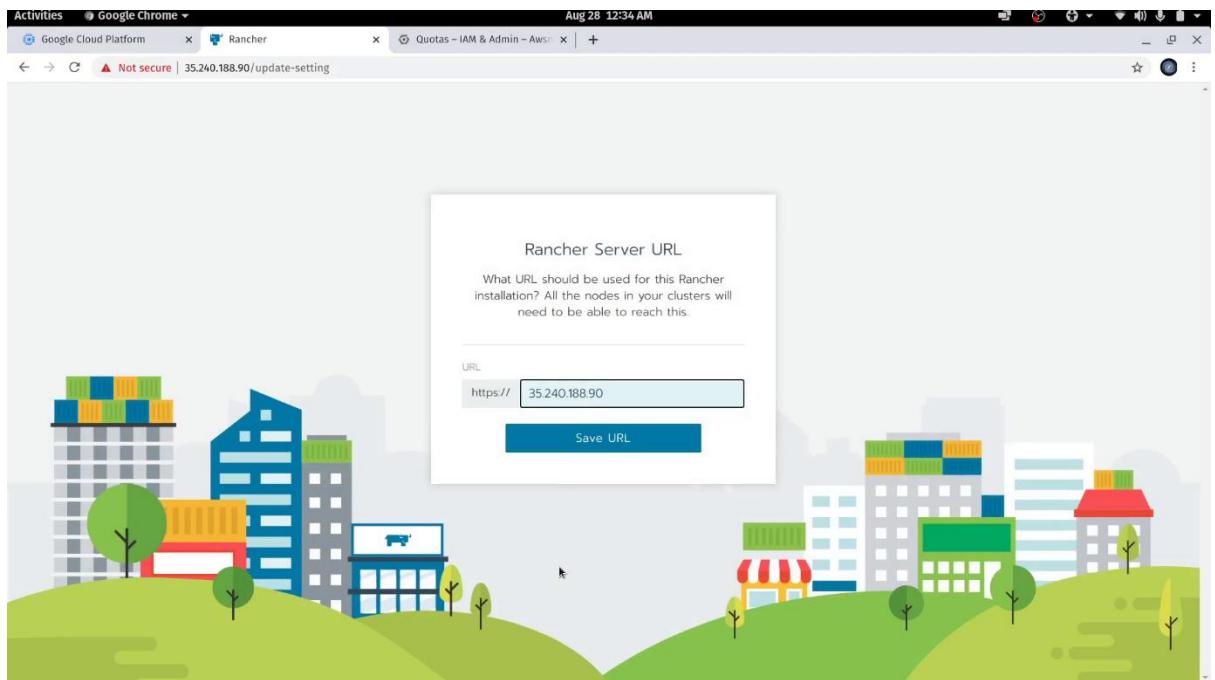
Note: Before setting up VM instances we need to create two different projects in Google Cloud Platform because each region has a quota for vCPU about 8 vCPUs. In one project we will setup Rancher, ELK Stack and Harness delegate and we will use the other project solely for Kubernetes cluster.

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.

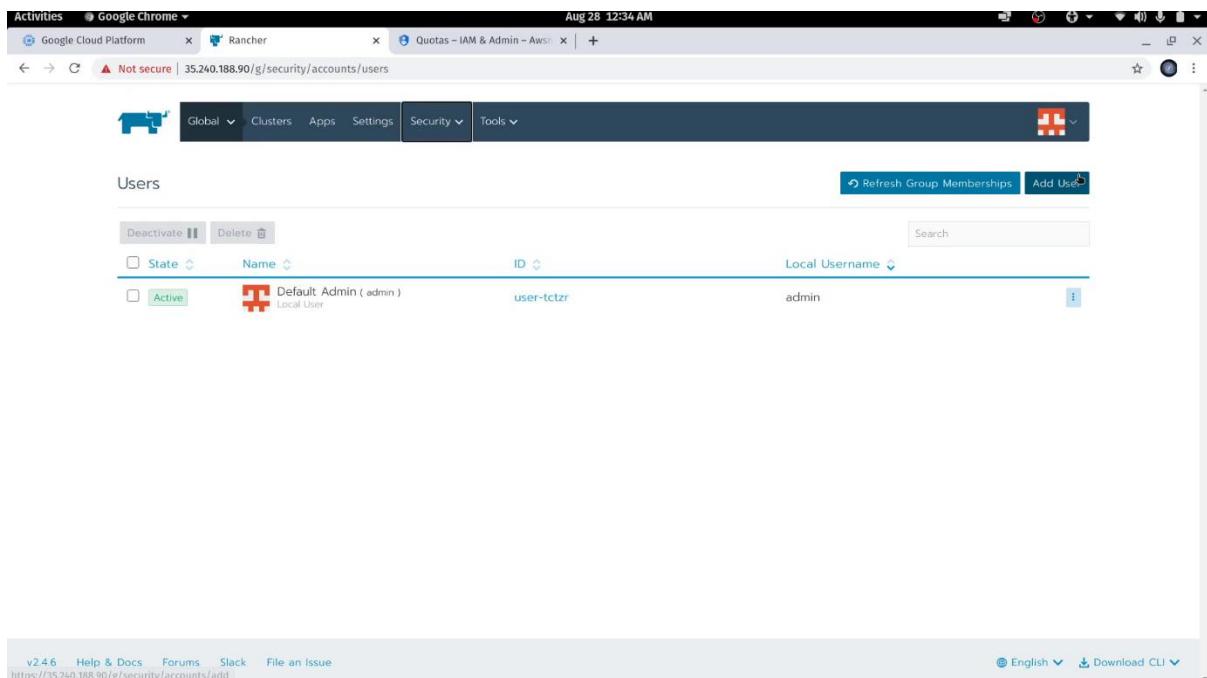


9. Save Rancher Server URL



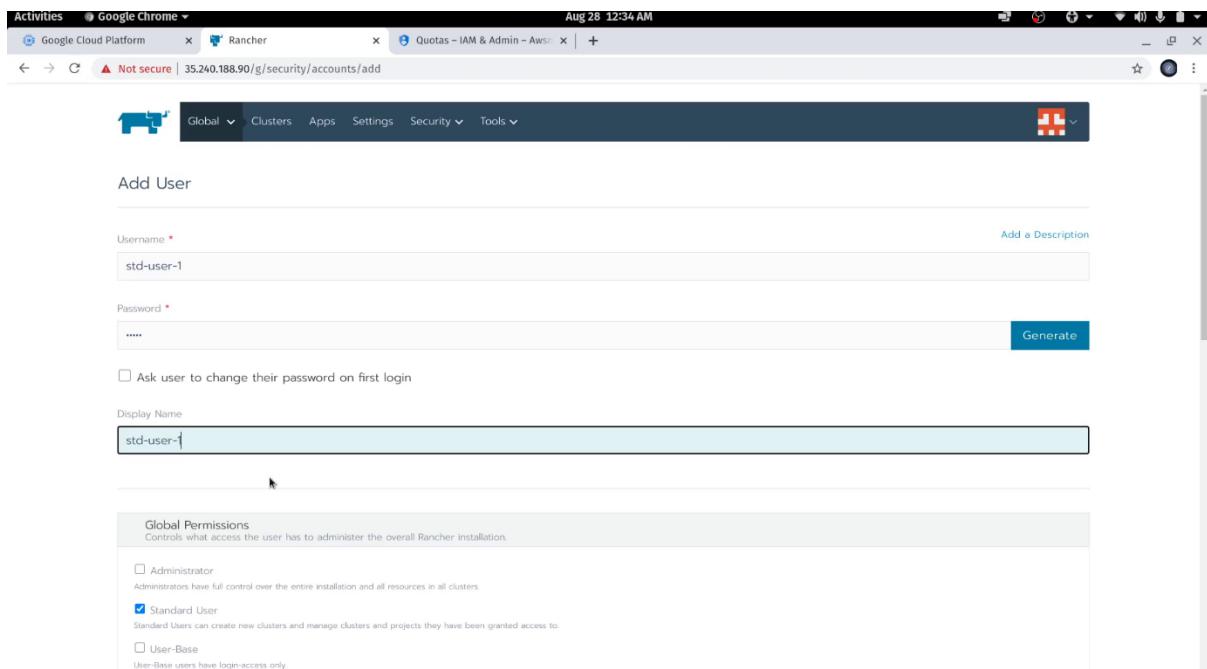
10. Now before we go ahead and create a cluster, let's see how to add a user and implement RBAC (Role Based Access Control) with Rancher.

- 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 bottom of the screen shows a navigation bar with links like 'Help & Docs', 'Forums', 'Slack', and 'File an Issue'.

- 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'. Under 'Global Permissions', the 'Standard User' checkbox is checked, while 'Administrator' and 'User-Base' are unchecked. A 'Generate' button is also visible.

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

The screenshot shows the 'Global Permissions' configuration page in Rancher. It includes the following sections:

- Global Permissions**: Controls what access the user has to administer the overall Rancher installation. Options include:
 - Administrator: Administrators have full control over the entire installation and all resources in all clusters.
 - Standard User: Standard Users can create new clusters and manage clusters and projects they have been granted access to.
 - User-Base: User-Base users have login-access only.
- Custom**: Roles not created by Rancher.
- Built-in**: Additional roles to define more fine-grain permissions model. Options include:
 - Create new Clusters: Allows the user to create new clusters and become the owner of them. Standard Users have this permission by default.
 - Create RKE Template Revisions: No description provided.
 - Create new RKE Cluster Templates: Allows the user to create new RKE cluster templates and become the owner of them.
 - Configure Authentication: Allows the user to enable, configure, and disable all Authentication provider settings.
 - Configure Catalogs: Allows the user to add, edit, and remove Catalogs.
 - Create new Cluster Drivers: Allows the user to create new cluster drivers and become the owner of them.

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' interface in Rancher. It includes the following sections:

- From existing nodes (Custom)**: Create a new Kubernetes cluster using RKE, out of existing bare-metal servers or virtual machines.
- Import an existing cluster**: Import an existing Kubernetes cluster. For EKS backed clusters, Rancher can manage some aspects of the cluster configuration, such as version upgrades. For standard Kubernetes clusters, the provider will manage provisioning and configuration.
- With RKE and new nodes in an infrastructure provider**:
 - Amazon EC2**
 - Azure**
 - DigitalOcean**
 - Linode**
 - vSphere**
- With a hosted Kubernetes provider**:
 - Amazon EKS**
 - Azure AKS**
 - Google GKE**

At the bottom right is a 'Cancel' button.

- 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
Local User	
std-user-1	Member

+ Add Member

Labels & Annotations None

Service Account *

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"

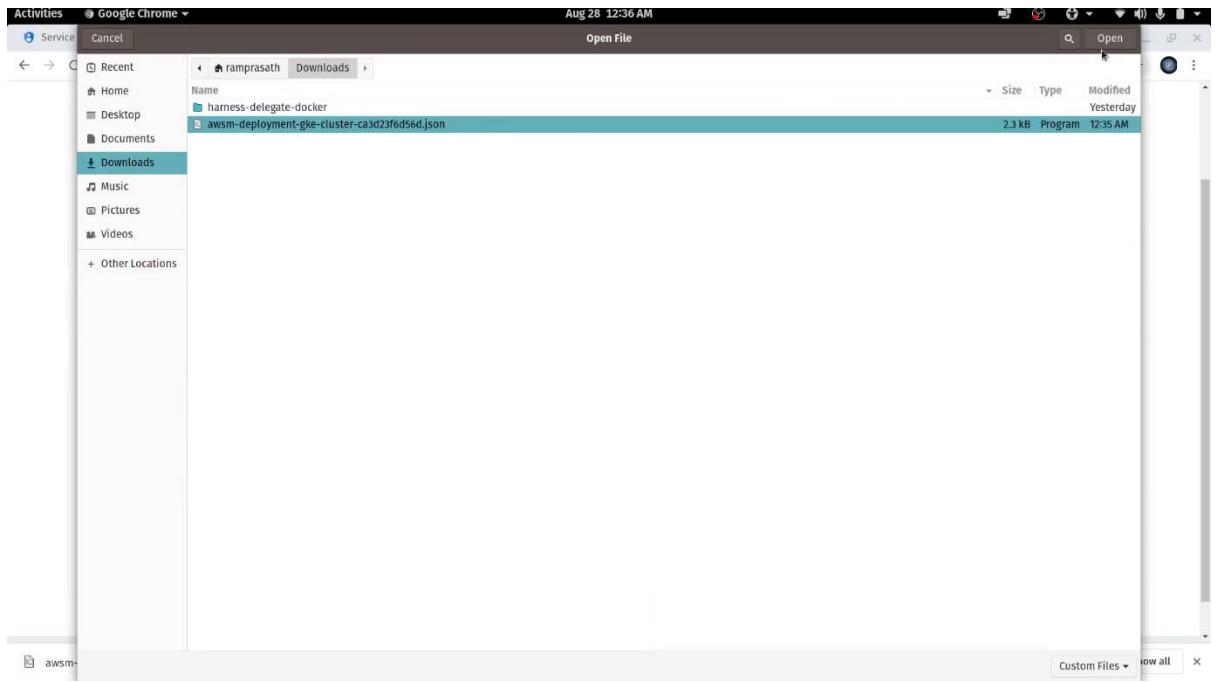
Email	Status	Name	Description	Key ID	Key creation date	Actions
compute@developer.gserviceaccount.com	Green	Compute Engine default service account	No keys			⋮

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 area and a "Read from a file" button. Below the input area is a note about required IAM roles: Compute Viewer, 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 web interface. The user is creating a new service account named "std-user-1" and selecting "Member" as the role. In the "Service Account" field, there is a large redacted JSON key. A warning message box is displayed, 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". At the bottom, there are "Next: Configure Nodes" and "Cancel" buttons. The page footer includes links for v2.4.6, Help & Docs, Forums, Slack, and File an Issue, along with language and download options.

- 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 API type, last update date, and service name. A file download button for 'awsm-deployment.json' is located 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. The page title is "Kubernetes Engine API". It includes a "Google" logo, a brief description ("Builds and manages container-based applications, powered by the open source Kubernetes technology."), and two buttons: "ENABLE" and "TRY THIS API". Below these buttons is a link "Click to enable this API". At the bottom of the main content area are tabs for "OVERVIEW", "PRICING", and "DOCUMENTATION". On the right side, there is a sidebar with sections for "Additional details" (Type: APIs & services, Last updated: 12/10/19, Category: Compute, Service name: container.googleapis.com) and a "Show all" link.

- 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/goglegke. The page title is "Add/Launch". It features a "INVITE YOUR TEAM" section with a "Default Admin (admin)" entry (Cluster Owner role) and a "std-user-1" entry (Member role). Below this is a "Service Account" section with a "Read from a file" button and a large redacted JSON key field. A note at the bottom of this section states: "Create a Service Account with a JSON private key and provide the JSON here. See Google Cloud docs for more info about creating a service account. These IAM roles are required: Compute Viewer [roles/compute.viewer], (Project) Viewer [roles/viewer], Kubernetes Engine Admin [roles/container.admin], Service Account User [roles/iam.serviceAccountUser]. More info on roles can be found here". At the bottom of the page, there is a "Authenticating..." progress bar and a "Cancel" button. The footer includes links for "v2.4.6 Help & Docs", "Forums", "Slack", "File an Issue", "English", "Download CLI", and "Show all".

- 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.

The screenshot shows the Rancher web interface. At the top, there's a navigation bar with tabs for 'Global', 'Clusters', 'Apps', 'Settings', 'Security', and 'Tools'. On the right side of the header, there's a 'Add Cluster' button. Below the header, the main content area is titled 'Clusters'. It features a table with columns: 'Delete', 'State', 'Cluster Name', 'Provider', 'Nodes', 'CPU', and 'RAM'. A single row is visible, representing the cluster 'aws-m-cluster'. The 'State' column shows 'Provisioning'. The 'Provider' column shows 'Google GKE'. The 'Nodes' column shows '0', 'CPU' shows 'n/a', and 'RAM' shows 'n/a'. There's also an 'i' icon in the last column of the row. At the bottom of the page, there's a footer with links for 'v2.4.6', 'Help & Docs', 'Forums', 'Slack', 'File an Issue', and language selection ('English') along with a 'Download CLI' link.

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

This screenshot shows the same Rancher interface after the cluster has been successfully provisioned. The 'Cluster Name' column now displays 'aws-m-cluster' with the 'State' set to 'Active'. All other details (Provider, Nodes, CPU, RAM) remain the same as in the previous screenshot. The footer at the bottom of the page includes links for 'v2.4.6', 'Help & Docs', 'Forums', 'Slack', 'File an Issue', and language selection ('English') along with a 'Download CLI' link.

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-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

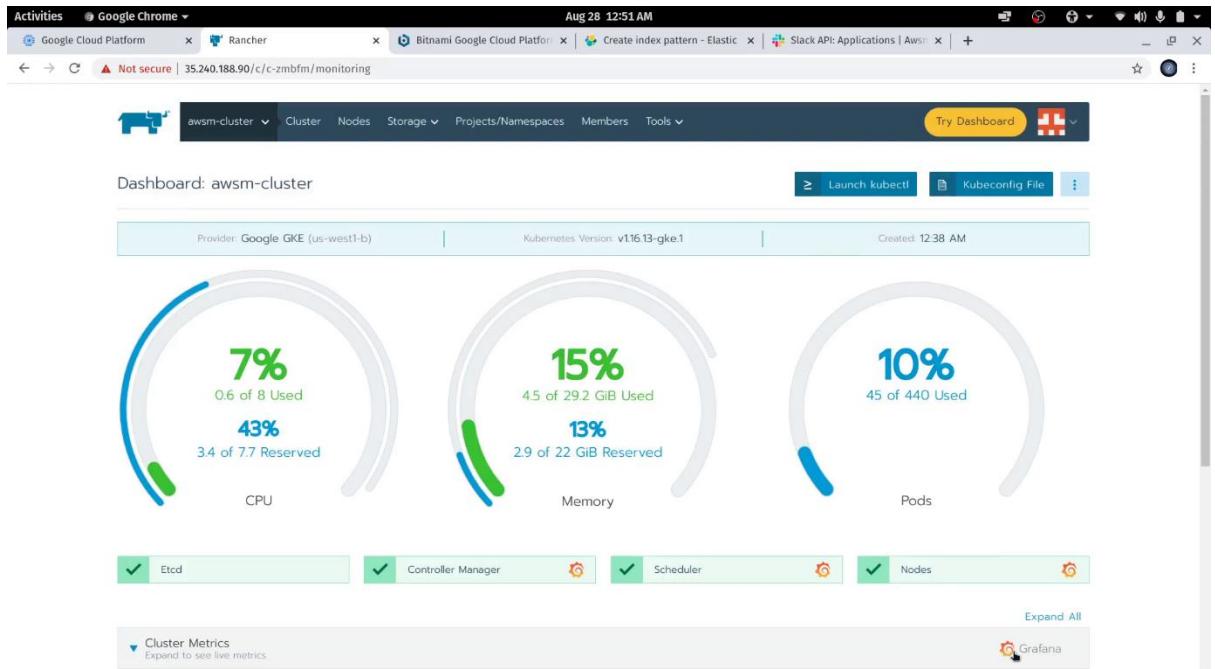
Enable Persistent Storage for Grafana:

True False

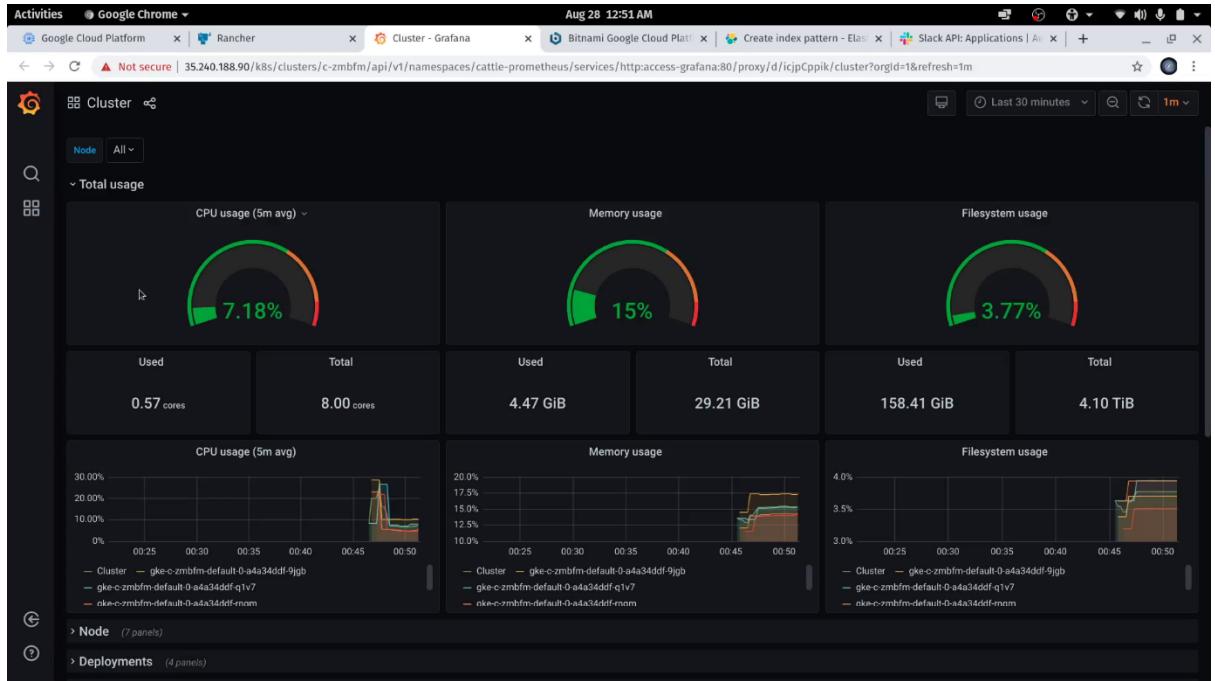
Show advanced options

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.

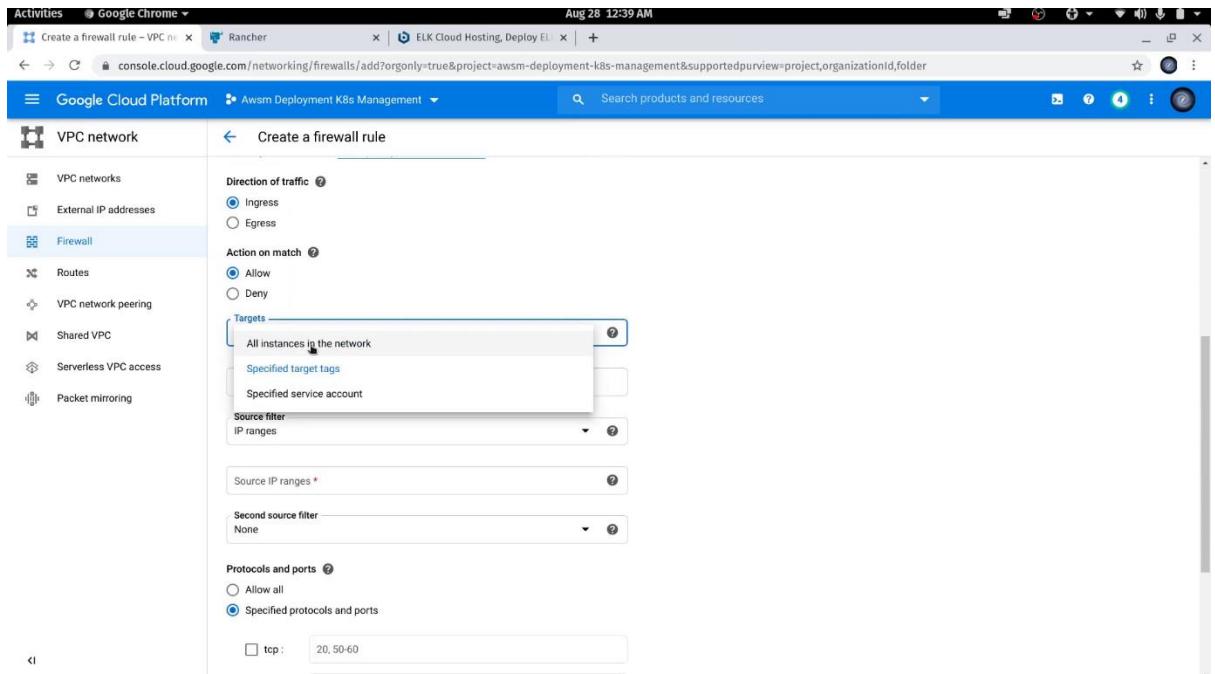


13. Now we will setup ELK Stack using Bitnami Launchpad for Google Cloud Platform.

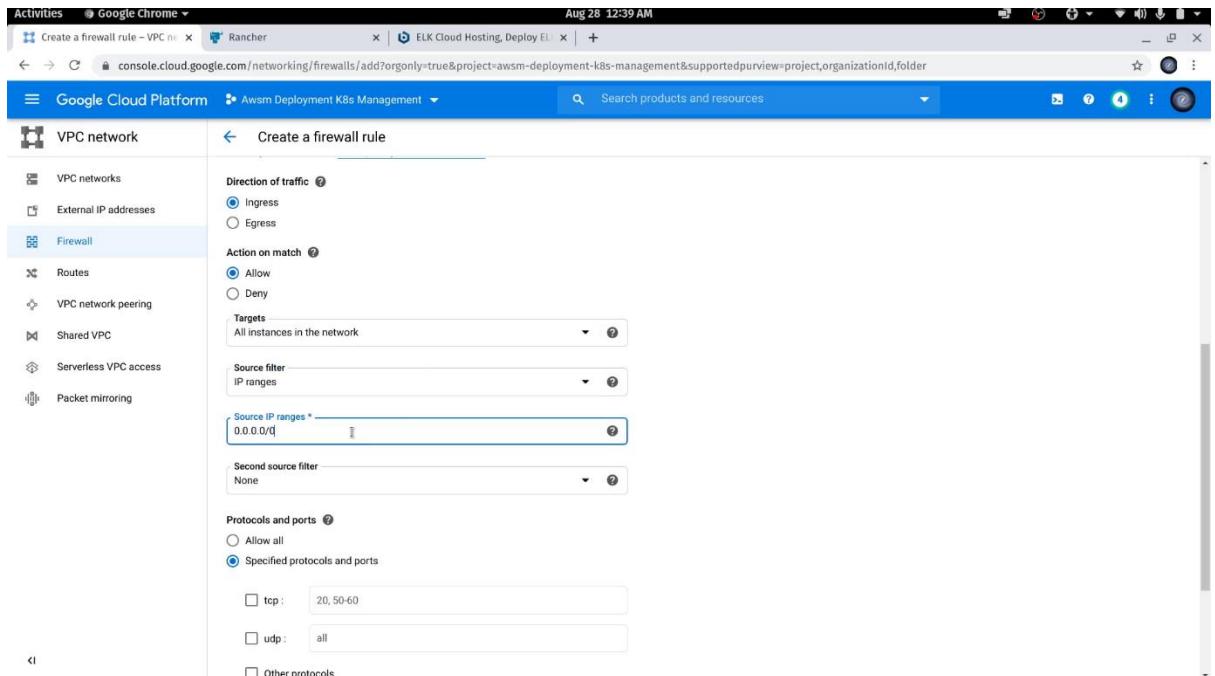
- Note: Before going to the link we need to add a firewall rule in GCP to allow TCP port 9200 for Elasticsearch.

The screenshot shows the "Create a firewall rule" page in the Google Cloud Platform interface. The sidebar on the left lists VPC network components: VPC networks, External IP addresses, Firewall (selected), Routes, VPC network peering, Shared VPC, Serverless VPC access, and Packet mirroring. The main form has fields for "Name" (set to "awsrm-elasticsearch"), "Description", "Logs" (set to "Off"), "Network" (set to "default"), "Priority" (set to "1000"), "Direction of traffic" (set to "Ingress"), and "Action on match" (set to "Allow"). A note at the bottom states: "Priority can be 0 - 65535 Check priority of other firewall rules".

- 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.'"/>

- **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 Add 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:

- 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 Awsm 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

ELK

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](#)

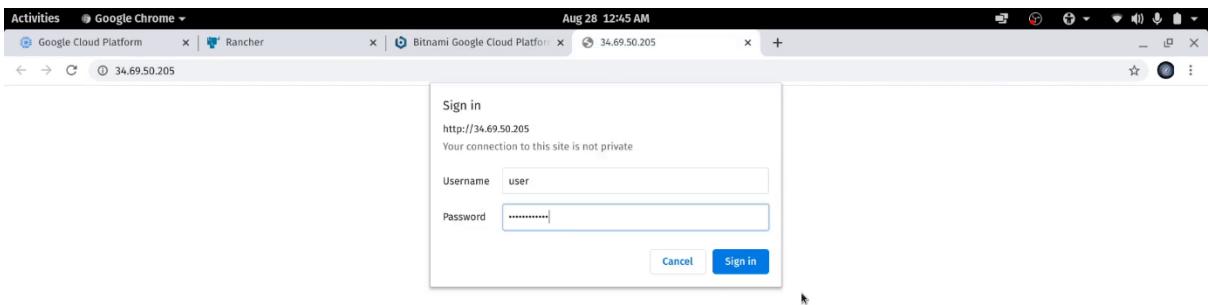
- We can verify the launch of ELK instance from GCP under the respective chosen project.

The screenshot shows the Google Cloud Platform Compute Engine interface. On the left, a 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 for Name, Zone, Recommendation, In use by, Internal IP, External IP, and Connect. Two instances are listed: 'awsm-rancher-server' in 'asia-southeast1-b' with IP 10.148.0.2 and 'bitnami-elk-7fb8' in 'us-central1-c' with IP 10.128.0.2. Below the table, there are related actions such as View Billing Report, Monitor VMs, Explore VM Logs, Setup Firewall Rules, and Patch Management.

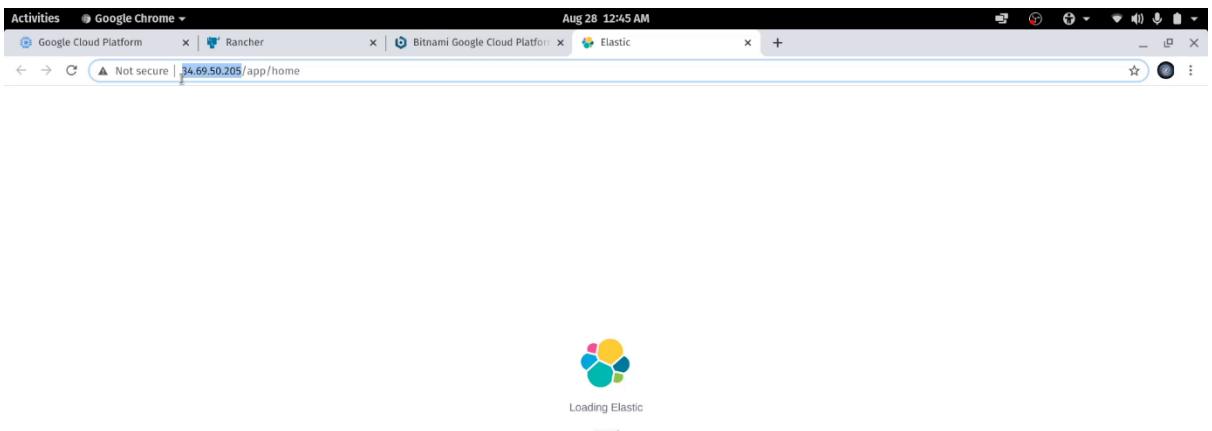
- Once the instance is created, click **GO TO APPLICATION**

The screenshot shows the Bitnami Launchpad interface for a Bitnami ELK stack. The top navigation bar includes links for VIRTUAL MACHINES, LIBRARY, SUPPORT, and ACCOUNT. The main content 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 stack, and two blue 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 provides details about the running instance: IP 34.69.50.205, Region US-CENTRAL1-C, and a monthly cost of \$36.67. It includes a 'LAUNCH SSH CONSOLE' button, download keys for PEM and PPK, and options to Show More, REBOOT, SHUTDOWN, or DELETE the instance.

- 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 dashboard for the awsm-cluster. At the top, there are three tabs: Google Cloud Platform, Rancher, and Bitnami Google Cloud Platform. Below the tabs, the URL is 35.240.188.90/c/c-zmbfm/monitoring. The main header includes the cluster name 'awsm-cluster' and a 'Tools' dropdown menu with options like Alerts, Catalogs, Notifiers, Logging (which is selected), Monitoring, Istio, and CIS Scans. A message 'Monitoring API is not ready' is displayed. Below the header, there are three circular progress bars: 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). Underneath the bars, a list of healthy services is shown: Etcd, Controller Manager, Scheduler, and Nodes. At the bottom, there is a section for 'Events' with a link to '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 in Rancher. The top navigation bar includes the cluster name 'awsm-cluster' and a 'Tools' dropdown menu. Below the navigation, there is a note: 'We will use fluentd to collect stdout/stderr logs from each container and the log files which exist under path /var/log/* on each host. The logs can be shipped to a target you configure below.' Below this note, there are six icons representing different log collectors: None (disabled), Elasticsearch (selected), Splunk, Kafka, Syslog, and Fluentd. A message 'Logging is disabled in the current cluster.' is displayed. At the bottom, there is a 'Save' button. The footer of the page 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.

Cluster Logging

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.

- 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": "WD68LUuhWVf5LMjqlh",
  "_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

- 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 below says 'Your index pattern matches 1 source.' A list shows 'awsm-cluster-2020-08-27' with an 'Index' button next to it. At the bottom is a 'Rows per page: 10' dropdown and 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 shows '@timestamp' at the top, which is highlighted with a blue background. Below it is a link '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

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

awsm-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

awsm-cluster-2020-08-27

Count

6,316 hits Aug 28, 2020 @ 00:38:25.861 - Aug 28, 2020 @ 00:53:25.861 Auto

@timestamp per 30 seconds

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.

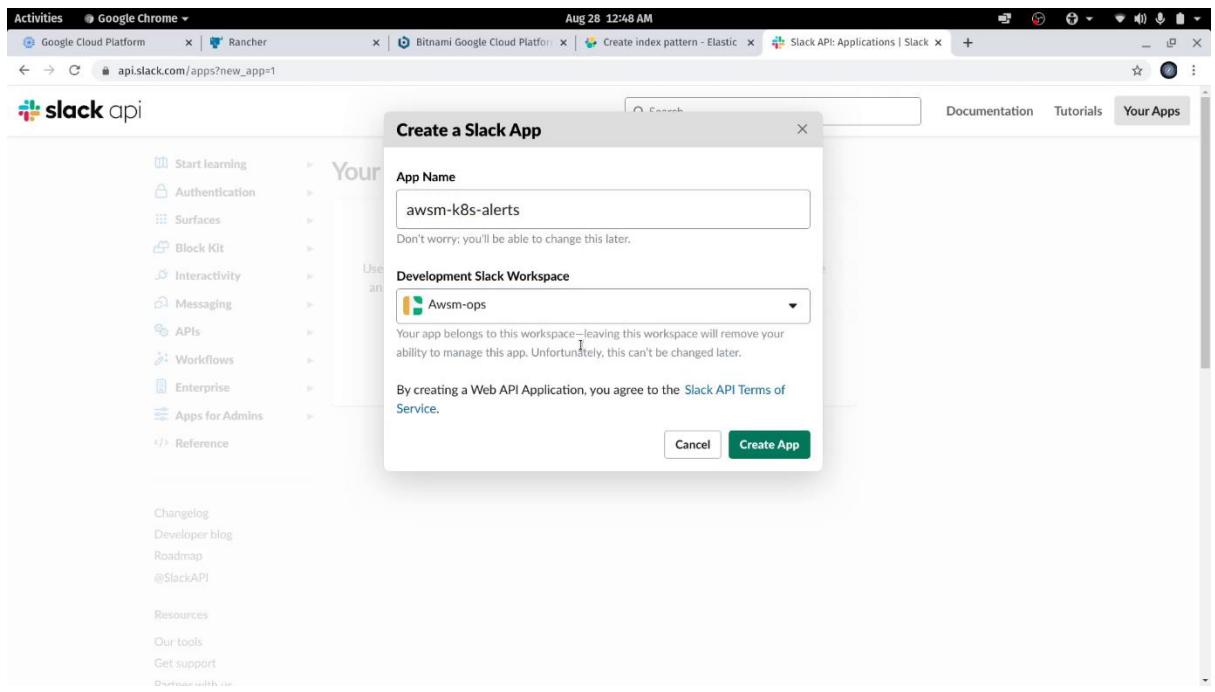
- Create a Slack workspace.
- Create a channel.
- Then go to <https://api.slack.com> and click **Start Building**.

The screenshot shows the Slack API homepage. At the top, there's a navigation bar with tabs for 'Documentation', 'Tutorials', 'Twitter', and a green 'Go to Slack' button. Below the navigation is a large, colorful graphic with the word 'Build' and 'internal tools' in the center, surrounded by various colored lines and shapes. A prominent green 'Start Building' button is located in the center of the graphic. To the left of the graphic is a sidebar with links to 'Start learning', 'Authentication', 'Surfaces', 'Block Kit', 'Interactivity', 'Messaging', 'APIs', and 'Workflows'. On the right side, there's a search bar with placeholder text 'Search our docs, tutorials, changelogs, and more', a 'Search' button, and a 'Send messages' section. The 'Send messages' section includes a message card for 'Visitbot' from 'Jimmy West' at 4:45PM, with fields for 'Purpose of visit:' (Meetup), 'Who are you here to see?:' (Denver), and links to 'messages & incoming webhooks'. A status message 'Waiting for px.spiceworks.com...' is visible at the bottom left.

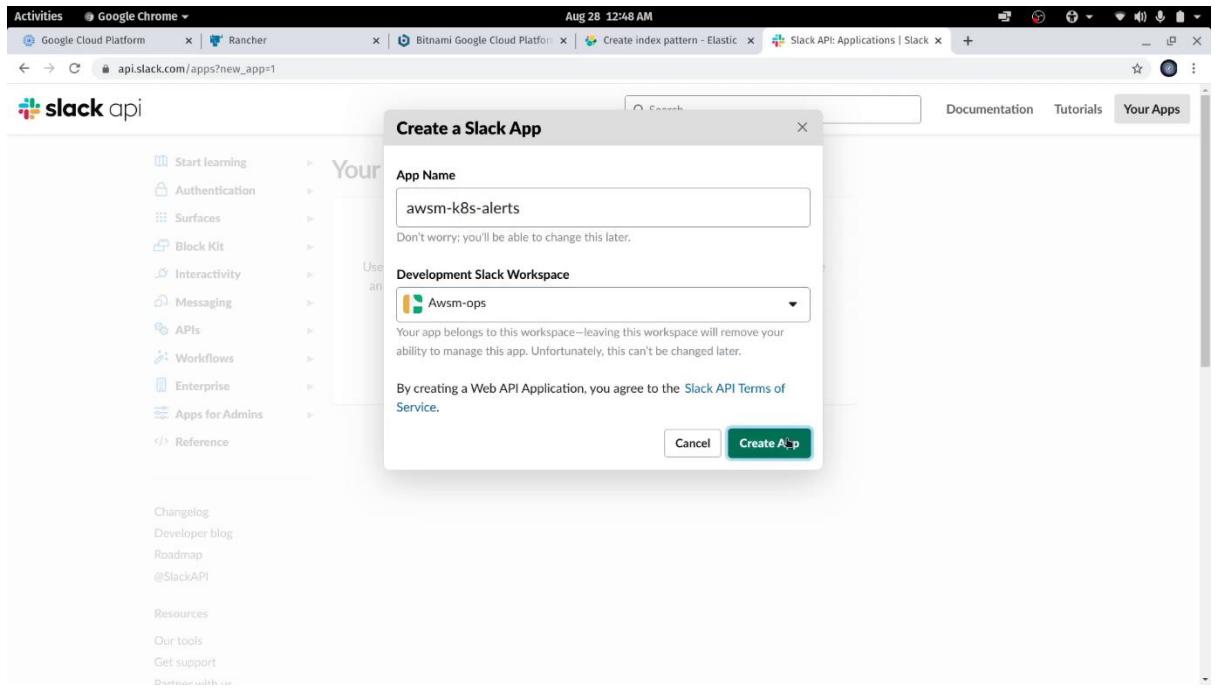
- Click on **Create an App**.

The screenshot shows the 'Your Apps' page. At the top, there's a navigation bar with tabs for 'Documentation', 'Tutorials', and a green 'Your Apps' button. Below the navigation is a sidebar with links to 'Start learning', 'Authentication', 'Surfaces', 'Block Kit', 'Interactivity', 'Messaging', 'APIs', 'Workflows', 'Enterprise', 'Apps for Admins', and 'Reference'. The main content area features a heading 'Your Apps' and a sub-section 'Build something amazing.' with text about using APIs to build apps. A large green 'Create an App' button is centered in this section. Below the main content, there's a note: 'Don't see an app you're looking for? Sign in to another workspace.' At the bottom of the page, there are links to 'Changelog', 'Developer blog', 'Roadmap', '@SlackAPI', 'Resources', 'Our tools', 'Get support', and 'Partner with us'.

- 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 bot users, 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.

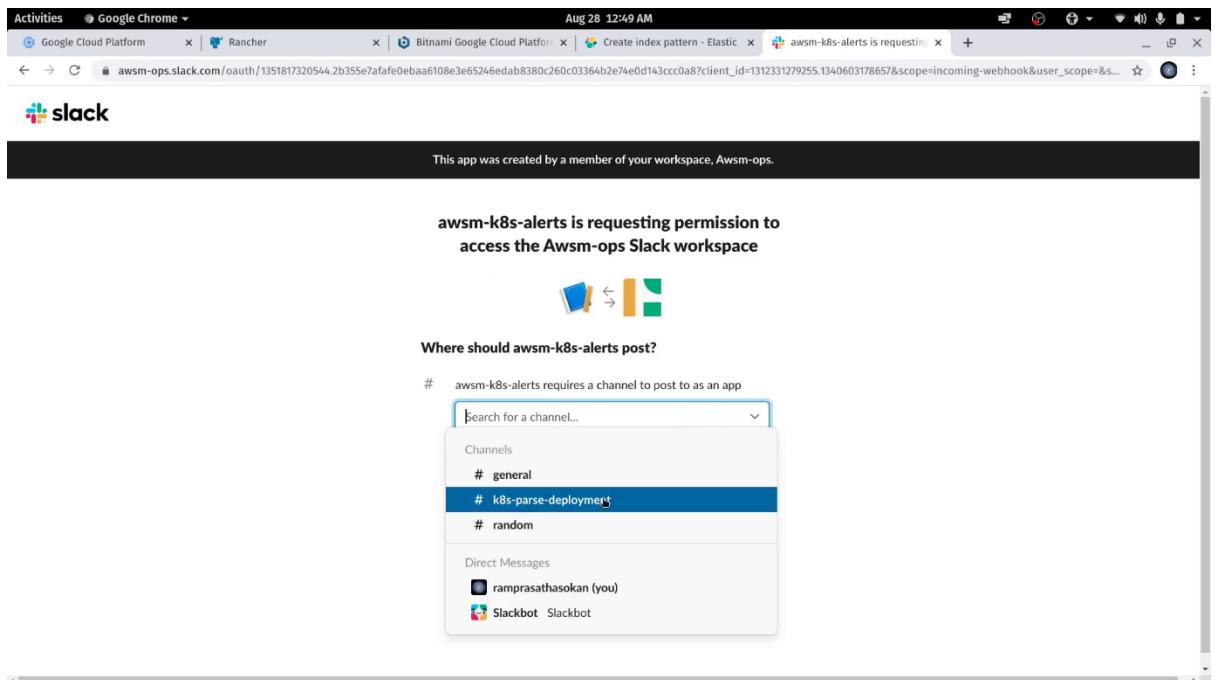
The screenshot shows the Slack API Applications interface. The left sidebar has a tree view with 'awsm-k8s-alerts' expanded, showing 'Basic Information', 'Collaborators', 'Install App', 'Manage Distribution', and 'Submit to App Directory'. Under 'Features', 'Incoming Webhooks' is selected, while other options like 'Interactivity & Shortcuts', 'Slash Commands', 'Workflow Steps', 'OAuth & Permissions', 'Event Subscriptions', 'User ID Translation', and 'Where's Bot User' are visible but not selected. Below the sidebar is a search bar and links for 'Documentation', 'Tutorials', and 'Your Apps'. The main content area is titled 'Incoming Webhooks' and contains sections for activating incoming webhooks (with a toggle switch set to 'On'), explaining what they are, and instructions for generating a new webhook URL. It also includes a 'Webhook URLs for Your Workspace' section with a curl command example and a table for managing webhook URLs.

- Add New Webhook to Workspace

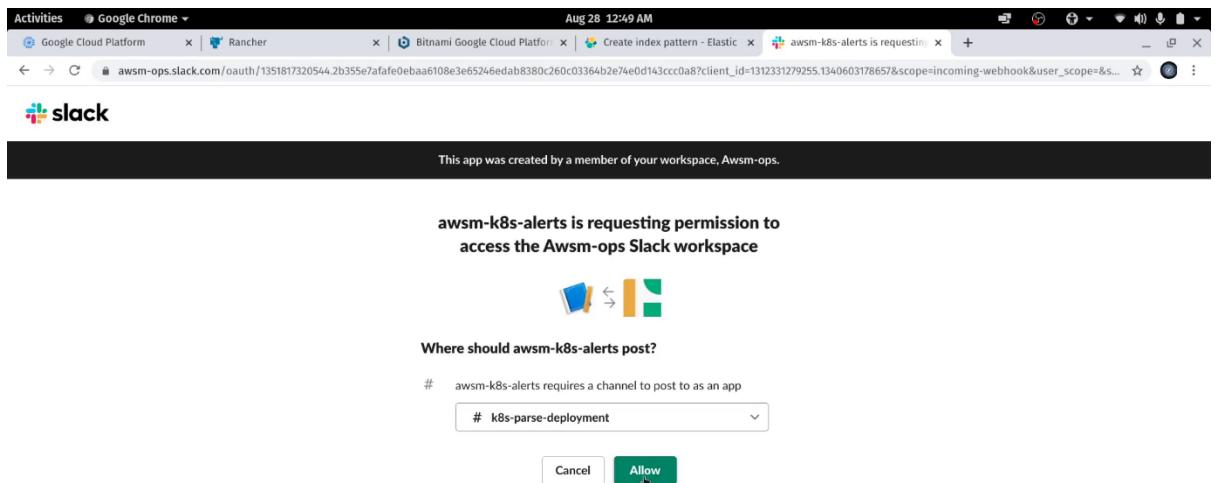
This screenshot is similar to the previous one but focuses on adding a new webhook. The 'Incoming Webhooks' section is active, and the 'Sample curl request to post to a channel:' field contains a curl command. Below it, a table lists 'Webhook URL', 'Channel', and 'Added By'. A note states 'No webhooks have been added yet.' At the bottom is a large red button labeled 'Add New Webhook to Workspace'.

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 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
https://hooks.slack.com/services/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	#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 donut charts show 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 the charts 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

Activities ● Google Chrome Aug 28 12:49 AM

Google Cloud Platform x Rancher x Bitnami Google Cloud Platform x Create index pattern - Elastic x Slack API: Applications | AWS x +

Not secure | 35.240.188.90/c/c-zmbfm/notifiers

Rancher awsm-cluster Cluster Nodes Storage Projects/Namespace Members Tools Try Dashboard

Notifiers Add Notifier

State Name Type Created

Search

There are no notifiers defined

v2.4.6 Help & Docs Forums Slack File an Issue English Download CLI

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

Activities ● Google Chrome Aug 28 12:49 AM

Google Cloud Platform x Rancher x Bitnami Google Cloud Platform x Create index pattern - Elastic x Slack API: Applications | AWS x +

Not secure | 35.240.188.90/c/c-zmbfm/notifiers

Rancher awsm Notifiers Add Notifier

Slack Email PagerDuty Webhook WeChat DingTalk Microsoft Teams

Name * awsm-k8s-alerts Add a Description

URL * https://hooks.slack.com/services/T01969R877H/B019N5DNLV9/fyHVmJBy8r7RaIhz8uWL1

Default Channel e.g. #example

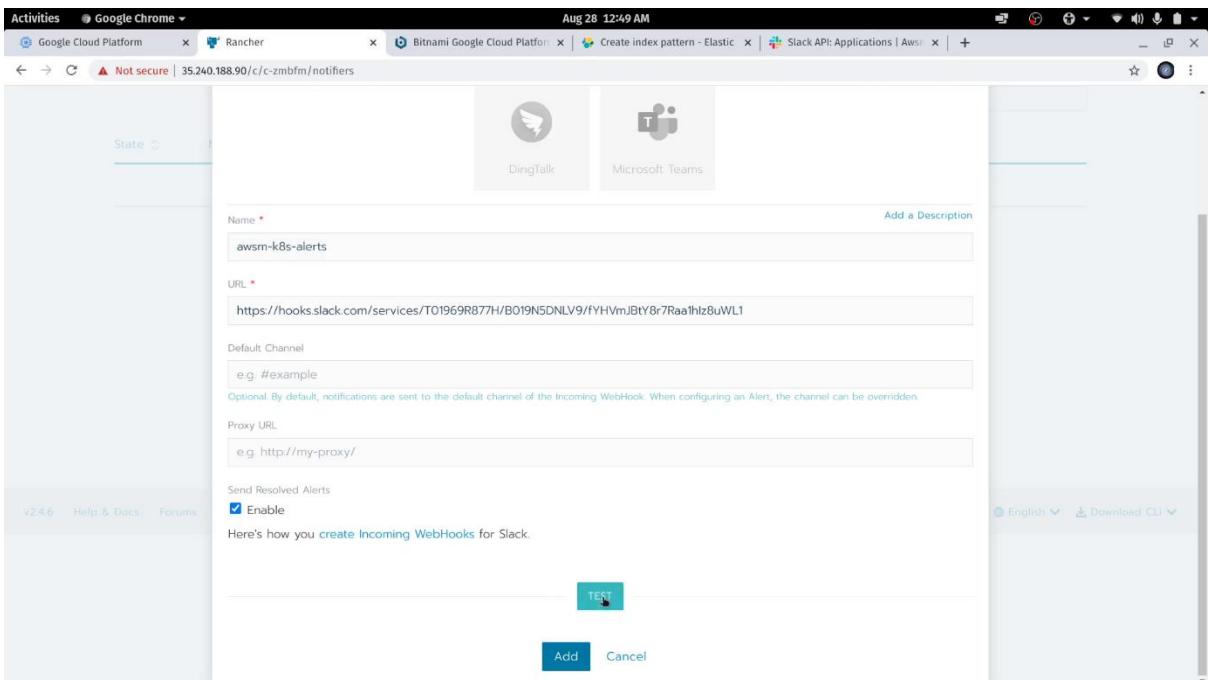
Optional. By default, notifications are sent to the default channel of the Incoming WebHook. When configuring an Alert, the channel can be overridden.

Proxy URL e.g. http://my-proxy/

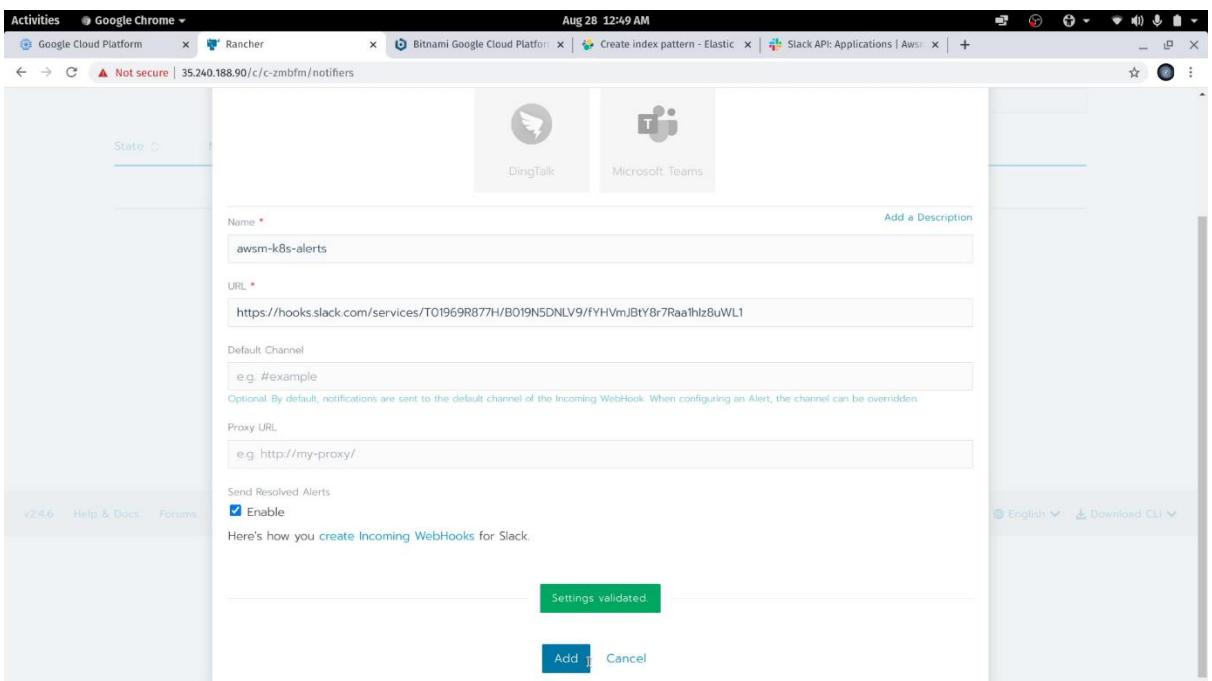
Send Resolved Alerts Enable

v2.4.6 Help & Docs Forums English Download CLI

- 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 single entry:

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

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

The screenshot shows the 'Alerts' section with multiple alert groups listed:

- Scheduler is unavailable**: Targeted at System Service, condition Unhealthy, not configured.
- Get warning deployment event**: Targeted at Deployment Event, happens, not configured.
- A set of alerts for node**: Alert for Node Memory, CPU, Disk Usage. Contains three rules:
 - High cpu load: Metric, Greater Than 100, not configured.
 - High node memory utilization: Metric, Greater Or Equal 80, not configured.
 - Node disk is running full within...: Metric, Not Null, not configured.
- A set of alerts for cluster scans**: Alert for Cluster Scans, both manual and scheduled. Contains two rules:
 - Manual Cluster Scan Completed: CIS Scan, Happens, not configured.
 - Manual Cluster Scan has Failures: CIS Scan, Failure, not configured.

- Now choose our newly added Slack notifier.

The screenshot shows a web browser window with multiple tabs open, including 'Google Cloud Platform', 'Rancher', 'Bitnami Google Cloud Platform', 'Create index pattern - Elastic', and 'Slack API: Applications | AWS'. The main content area displays a chart with numerical values on the y-axis (ranging from 0 to -5000000000) and time on the x-axis (Aug 28 12:49 AM to Aug 28 00:00). Below the chart, there are options to 'Send a' message type (Critical, Warning, Info), a 'Group Inherited' checkbox, and an 'Enabled' switch. A large blue button labeled '+ Add Alert Rule' is visible. Below this, the 'Alert' section shows 'To' set to 'Choose a Notifier' (with 'awsrn-k8s-alerts (Slack)' selected), 'Not Configured', and a note about overriding the value. A 'Show advanced options' link is present. At the bottom are 'Save' and 'Cancel' buttons, along with navigation links for 'v2.4.6 Help & Docs Forums Slack File an Issue' and language selection ('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**

The screenshot shows the same web browser interface as the previous one, but with the 'Show advanced options' link expanded. It displays three time configuration fields: 'Group Wait Time' (set to 0 days, 0 hours, 0 minutes, and 30 seconds), 'Group Interval Time' (set to 0 days, 0 hours, 0 minutes, and 30 seconds), and 'Repeat Interval Time' (set to 1 hour, 0 minutes, and 0 seconds). Below these fields, explanatory text describes their purposes. At the bottom are 'Save' and 'Cancel' buttons, along with navigation links for 'v2.4.6 Help & Docs Forums Slack File an Issue' and language selection ('English Download CLI').

- 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.

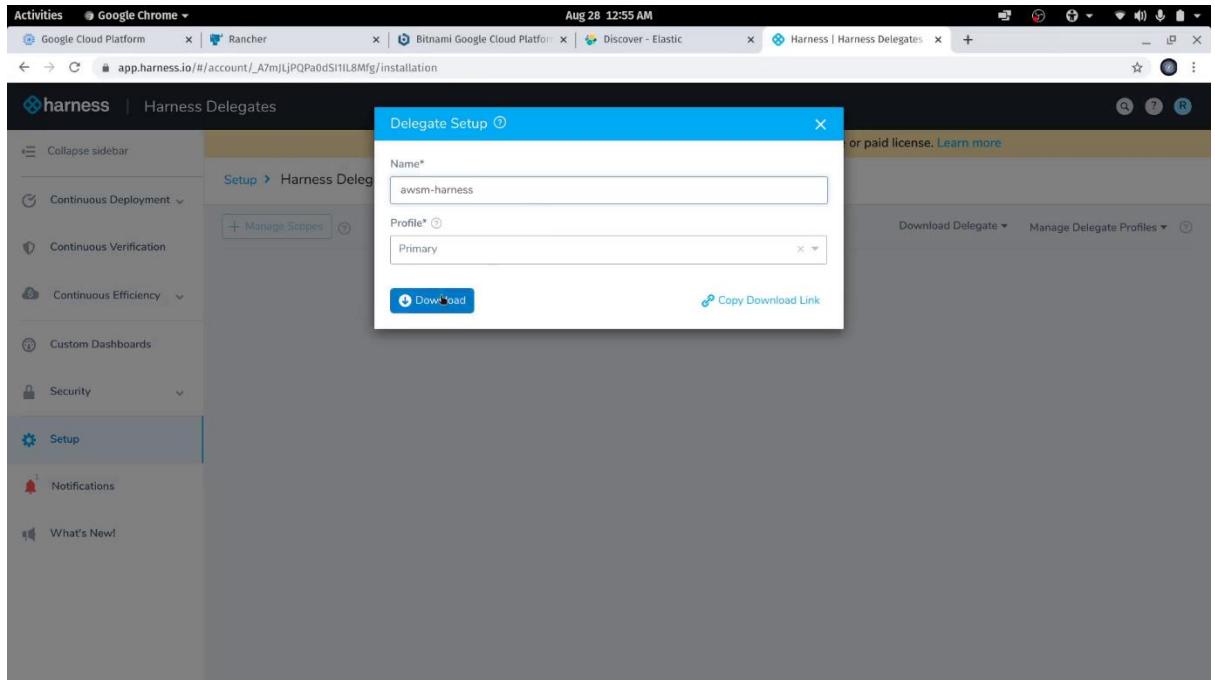
- After signing in, go to **Setup > Harness Delegates**

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.

- 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, including 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 variables are being processed by a pipeline and passed through sed commands to escape special characters.

- 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
Open README.txt launch-harness-delegate.sh
Save + ramprasadhasokan@aws-harness-delegate:~ Aug 28 1:01 AM
Instructions.txt README.txt launch-harness-delegate.sh
Aug 27 19:29:08 awsm-harness-delegate dockerd[3232]: time="2020-08-27T19:29:08"
Aug 27 19:29:08 awsm-harness-delegate dockerd[3232]: time="2020-08-27T19:29:08"
Aug 27 19:29:08 awsm-harness-delegate dockerd[3232]: time="2020-08-27T19:29:08"
Aug 27 19:29:08 awsm-harness-delegate systemd[1]: Started Docker Application C
Aug 27 19:29:08 awsm-harness-delegate dockerd[3232]: time="2020-08-27T19:29:08"
Lines 1-19/19. (END)
ramprasadhasokan@aws-harness-delegate:~$ sudo usermod -G docker ${USER}
ramprasadhasokan@aws-harness-delegate:~$ sudo su - ${USER}
ramprasadhasokan@aws-harness-delegate:~$ id -nG
ramprasadhasokan adm dialout cdrom floppy audio dip video plugdev lxd netdev d
ocker ubuntu google-sudoers
ramprasadhasokan@aws-harness-delegate:~$ sudo nano launch-harness-delegate.sh
ramprasadhasokan@aws-harness-delegate:~$ sudo chmod +x launch-harness-delegat
e.sh
ramprasadhasokan@aws-harness-delegate:~$ ./launch-harness-delegate.sh
latest: Pulling from harness/delegate
7595c8c21622: Pull complete
d13af8ca89f8: Pull complete
70799171ddba: Pull complete
b6c12202c5ef: Pull complete
4ef1facdef72: Pull complete
0b547720c433: Pull complete
d09008bf999: Pull complete
e723c5c5aa75: Pull complete
72f1fc1455c7: Pull complete
9028fb0ab03e2: Pull complete
0638a82af21f: Pull complete
5442cd1fe6c: Pull complete
Digest: sha256:22c8548b30a45b1612e247f08a71c4b99d0c70362c868309e37f875f851dd96
1
Status: Downloaded newer image for harness/delegate:latest
docker.io/harness/delegate:latest
5f27fe99d27e1c0aa32f06265abb416d012d23daf15b05f8cafafffc87bd7
ramprasadhasokan@aws-harness-delegate:~$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED          STATUS              PORTS               NAMES
5f27fe99d227        harness/delegate:latest "/bin/sh -c './start...'  22 seco
nds ago           Up 17 seconds          unruffled_mirzakhani
ramprasadhasokan@aws-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 Delegate setup interface. On the left, there's a sidebar with options like Continuous Deployment, Continuous Verification, Continuous Efficiency, Custom Dashboards, Security, Setup (which is selected), Notifications, and What's New!.

The main content area is titled "Setup > Harness Delegates". It displays a table for the "awsm-harness-delegate" delegate:

Name	awsm-harness
Hostname	awsm-harness-delegate
Description	<input type="text"/>
Delegate Type	DOCKER
Profile	<input checked="" type="radio"/> Primary
IP	172.17.0.2
Status	Connected
Last heartbeat	a few seconds ago - 08/28/2020 01:01 am
Active Versions	0.1.0.57604
Scope Limited To	+ Add Scope
Scope Excluded	+ Exclude Scope
Implicit Selectors	<input type="checkbox"/> awsm-harness
Selectors	<input type="checkbox"/> awsm-harness-delegate
	Edit

At the top of the page, there's a message: "Your Harness Professional trial will expire in 12 days. Keep going with a free or paid license. [Learn more](#)".

- 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. Inside the modal, there are 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, the main interface shows a list of delegates, with one entry for "awsmt-harness-delegate" which is currently set to "Primary".

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

The screenshot shows the Harness Delegates setup interface. The main list view shows the "awsmt-harness-delegate" entry, which has been updated to be the primary delegate. The "Delegate Type" column for this entry shows "Primary". Other columns include "Name" (awsmt-harness-delegate), "Hostname", "Description", "Profile", "IP", "Status", "Last heartbeat", "Active Versions", "Scope Limited To", "Scope Excluded", "Implicit Selectors", and "Selectors". A message at the top of the page says "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.

- 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...
      VMZ29F3SUJB20lCQURB50JnZ3foa2pPUFFREFqQtDnUhd3R2dZRFZRUJtFeE5rZVc1aGJxBoKY...
      kdsenRHnVwAh0jNkbk1sc3dHUVLVEVFRE45atLVv0YldsanJHbHpKR1Z1WlJ1dFkyRxdta...
      GNTwPbDwPREkzTVRr0d16TTxAGNTxpbBd09ESTFNVt3TxNPWVdqTdNuhd3R2dZRFZRUJtFe...
      E5rZVc1aGJxBoK1R2x6CmRHnVwAh0jNkbk1sc3dHUVLVEVFRE45atLVv0YldsanJHbHpKR1...
      Z1WlJ1dFkyRxdFkyxDxXVFU0mJcWhraK0UFJQKJnZ3foa2pPUFFN0KJ3TNbQ0W0M95KpG6y99z...
      1EyUhF2YlRETEUEyM5P0HFK2RLyS9l02tPVL1PygjY)NaVp35v1LYzcmaJ4V2dcz3BzHLr...
      m9ee1dUUvxdzJHQ2RckSPcW57xdJVEFP0m0dvhkRoEJBZJfFcK3BTUNbcVf3Rh2ZRFZSMFRBU...
      UgyOKFVd0f3RUiIveFL0mdnchra9QUVFEE0w0s0kFEQkdBaUvBdw0Rc03Ryak0k02FENz1RlwT...
      ThCbW0QcFJCNE1P25tjMG00VE00x0M0VDFSVFDnhMUXZXQ1FnU3J3wZvdUtyamdtc2l3VgpRY...
      3RwaiBKZ6pxSGxiYkZnEWc9P0gtLS0tLUIVRCB0RVJUSUZ3Q0UFURS0tLS0t"
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.

Add Kubernetes Cluster Cloud Provider

Display Name*

Awm Cluster

Cluster Details

Inherit from selected Delegate Enter manually

Master Url*

https://35.240.188.90/k8s/clusters/c-zmbfm

Authentication*

Select

Username and password

Service Account Token

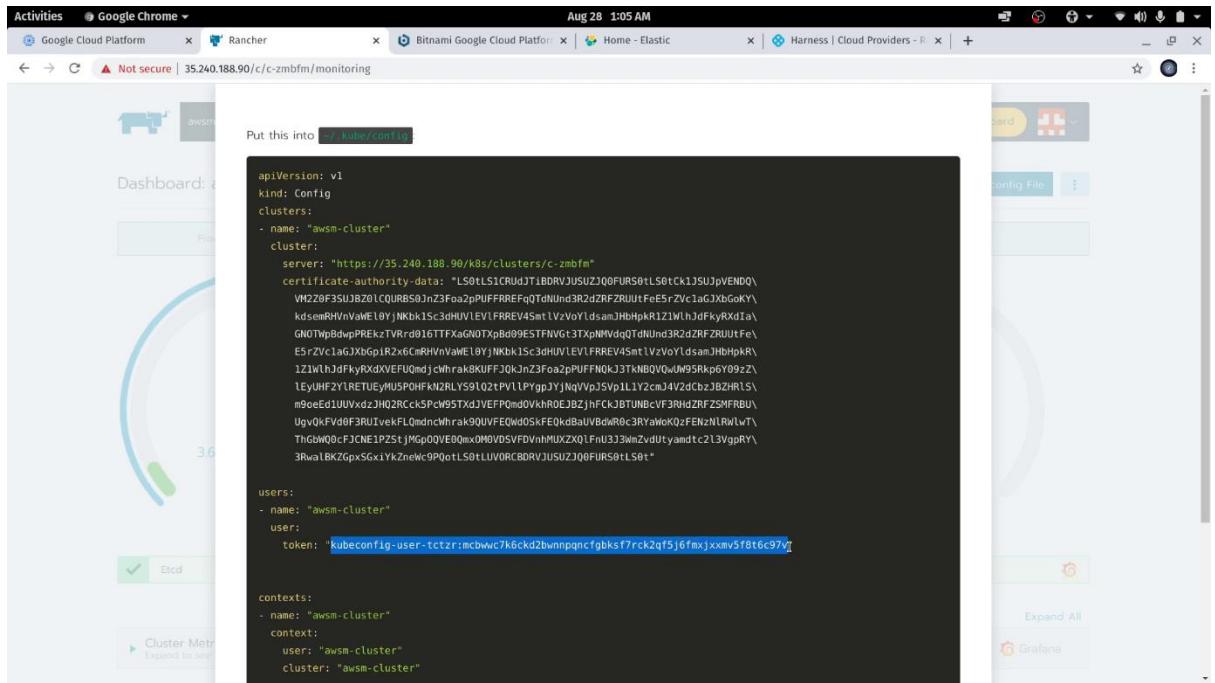
OIDC Token

Custom

All Applications	Production Environments
All Applications	Non-Production Environments

Test Submit

- Copy the Service Account Token from Kubeconfig file

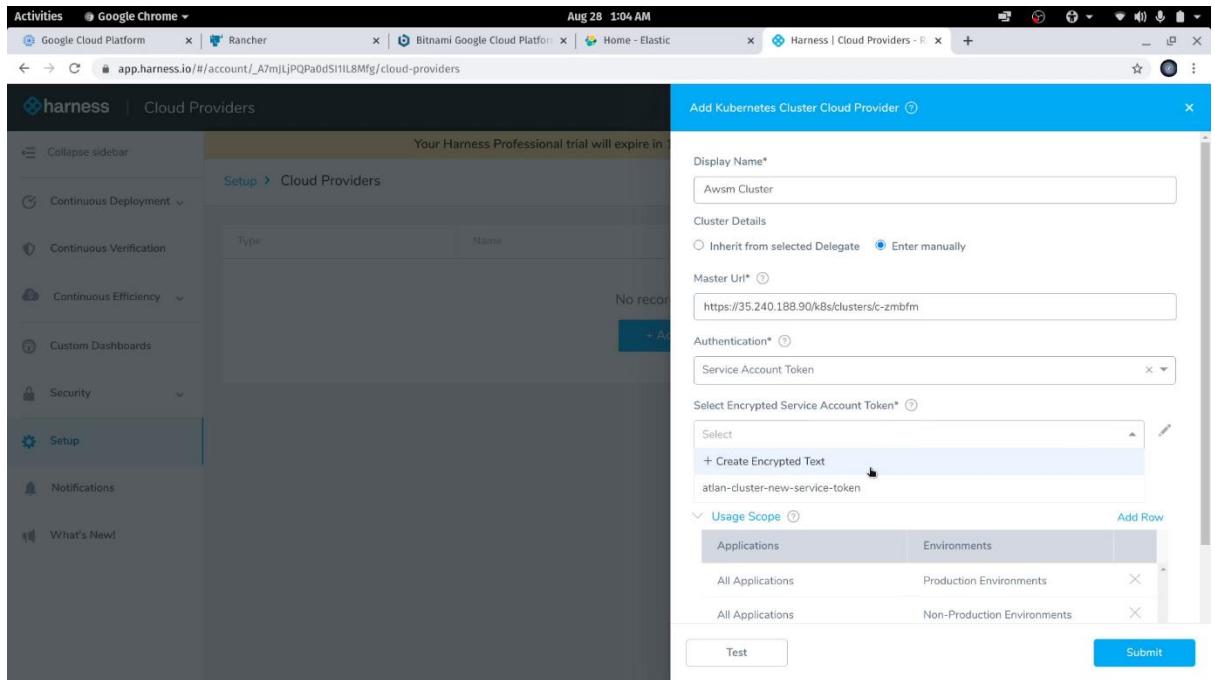


```

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...
      1EyUhF2YlRETEUEyUSP0HFK2RLyS9l02tPVL1PygpJY)NaVp35v1LYzcJ4V2cdzB2HLr...
      m9ee1dUUvxdzJHQ2RckSPcW57xdJVEFP0m0d0khROEBZJfFcK3BTUNbcV3RhHzFRBvU...
      UgyOKFVd0f3RUIveFL0mdnchra9QUVFEOw0s05kFEQkdBaUvBdw0c0c3Ryak0k02FENz1Rw...
      ThObmQ0cFJCNc1P25tjMG00VE00mx0M0VSDVFdnhMuxZXQlFnU3J3wZvdUyamdtc2l3Vg...
      3RwaiBKZ6pxSGxiYkZnEWc9P0gtLlIVORCB0RVJUSUZ3Q0FURS0tLS0t"
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*

Cluster Details

Inherit from selected Delegate Enter manually

Master Url*

Authentication*

Select Encrypted Service Account Token*

Select

+ Create Encrypted Text

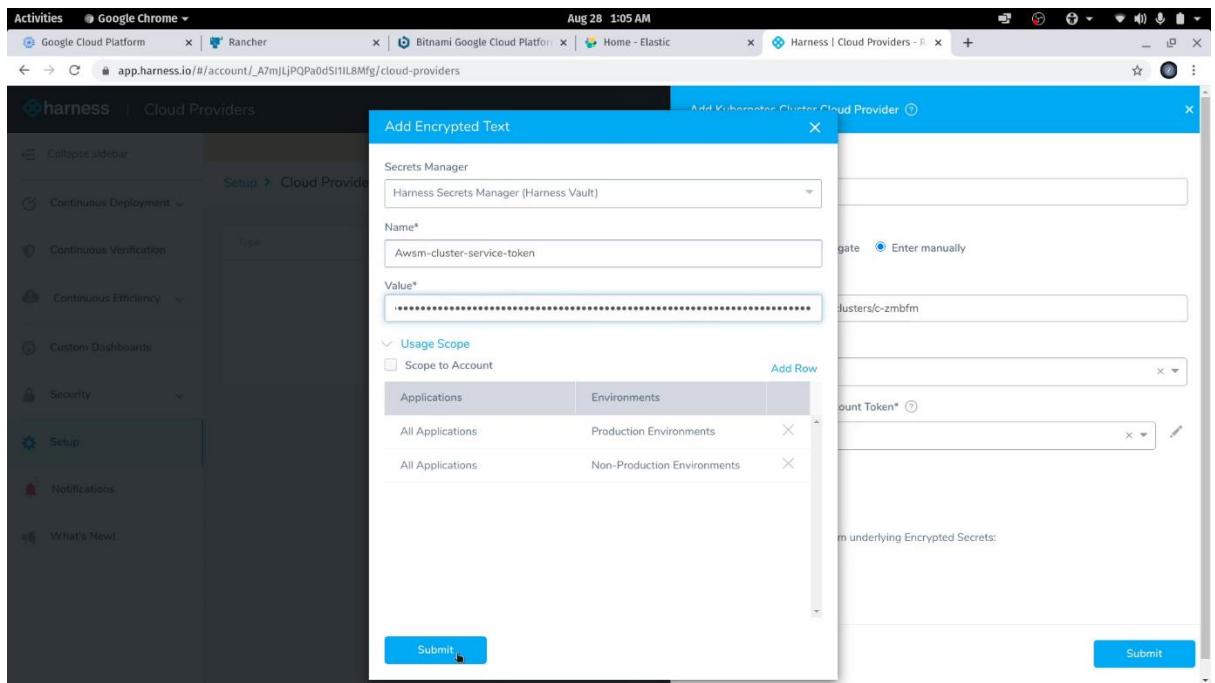
atlan-cluster-new-service-token

Usage Scope	
Applications	Environments
All Applications	Production Environments
All Applications	Non-Production Environments

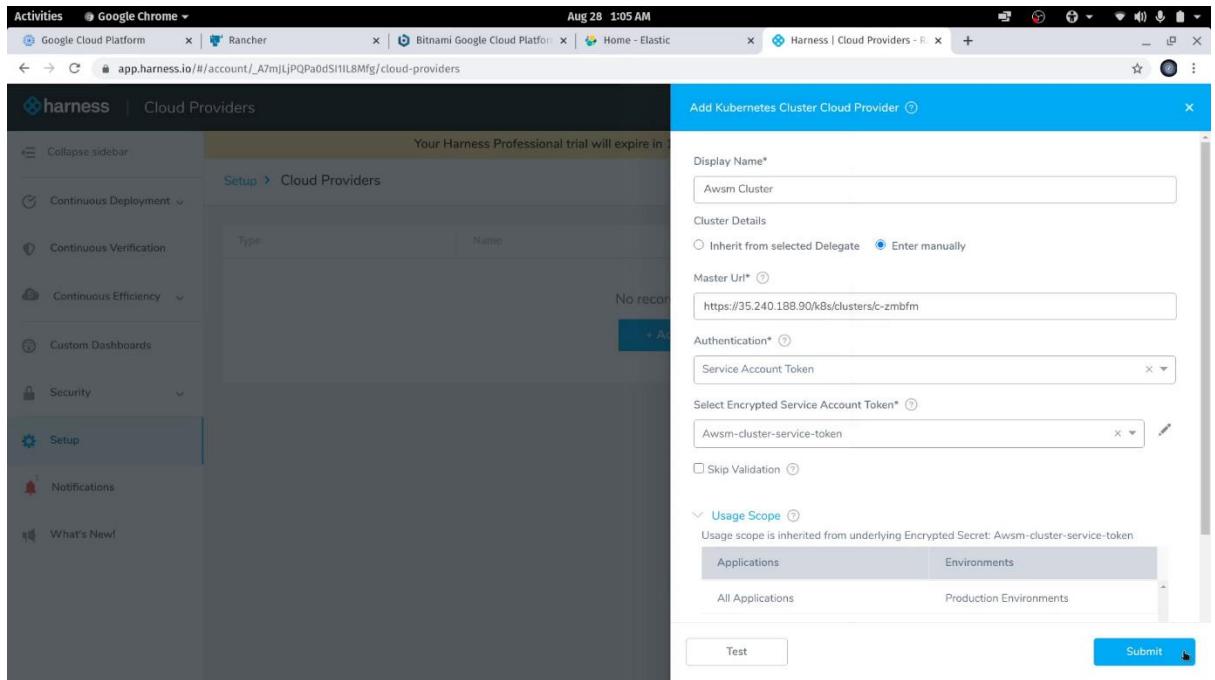
Test

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, 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. The main area displays a table titled 'Cloud Providers'. A single row is present, showing a Kubernetes icon, the name 'Awsm Cluster', and a URL 'https://35.240.188.90/k8s/clusters/c-zmbfm'. A message at the top indicates a 12-day trial expiration.

- 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, with the Setup option selected. The main area has a 'Setup' header. On the right, there's a sidebar with sections for Shared Resources (Cloud Providers, Connectors, Template Library, Application Stacks), Account (Tags Management, Alert Notification Rules, Harness Delegates), and a navigation bar with a URL 'https://app.harness.io/#/account/_A7mjLjPQP0dS11L8Mfg/connectors'.

- 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.

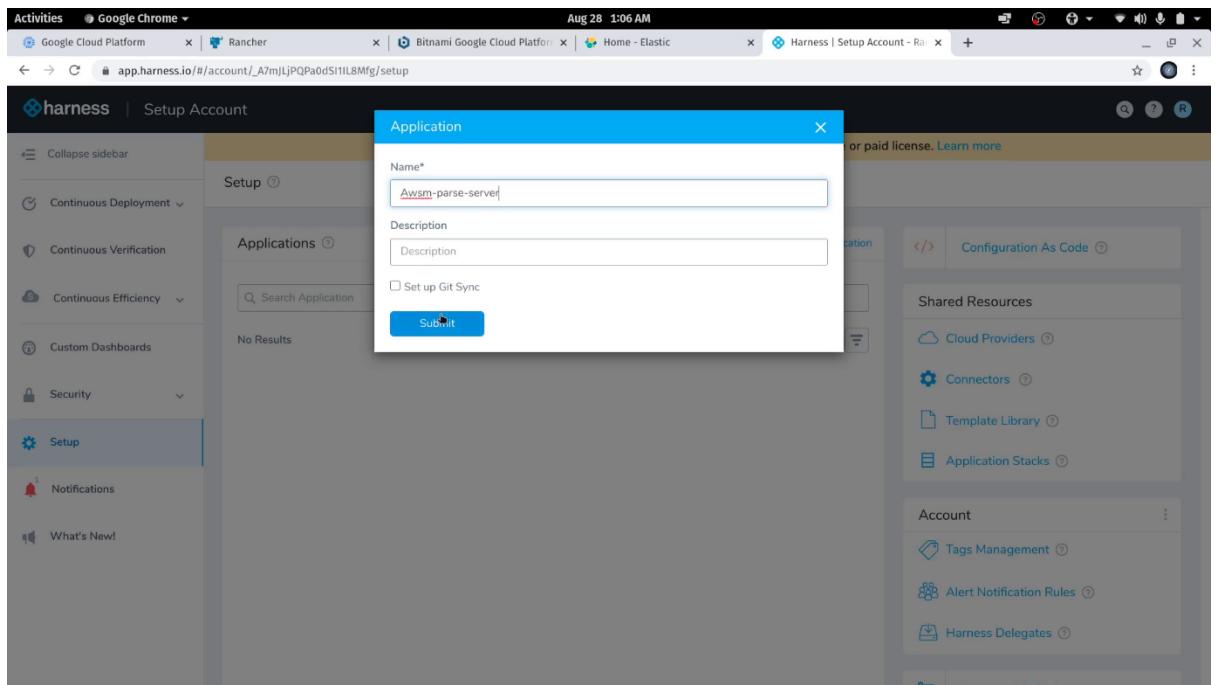
The screenshot shows the Harness web interface. The left sidebar is collapsed. The main navigation bar at the top includes tabs for Google Cloud Platform, Rancher, Bitnami Google Cloud Platform, Home - Elastic, and Harness | Artifact Servers. The current page is 'Artifact Servers' under 'Setup > Connectors'. 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	https://registry.hub.docker.com/v2/	⋮
Helm Repository	Bitnami Helm Hub	https://charts.bitnami.com/bitnami	⋮

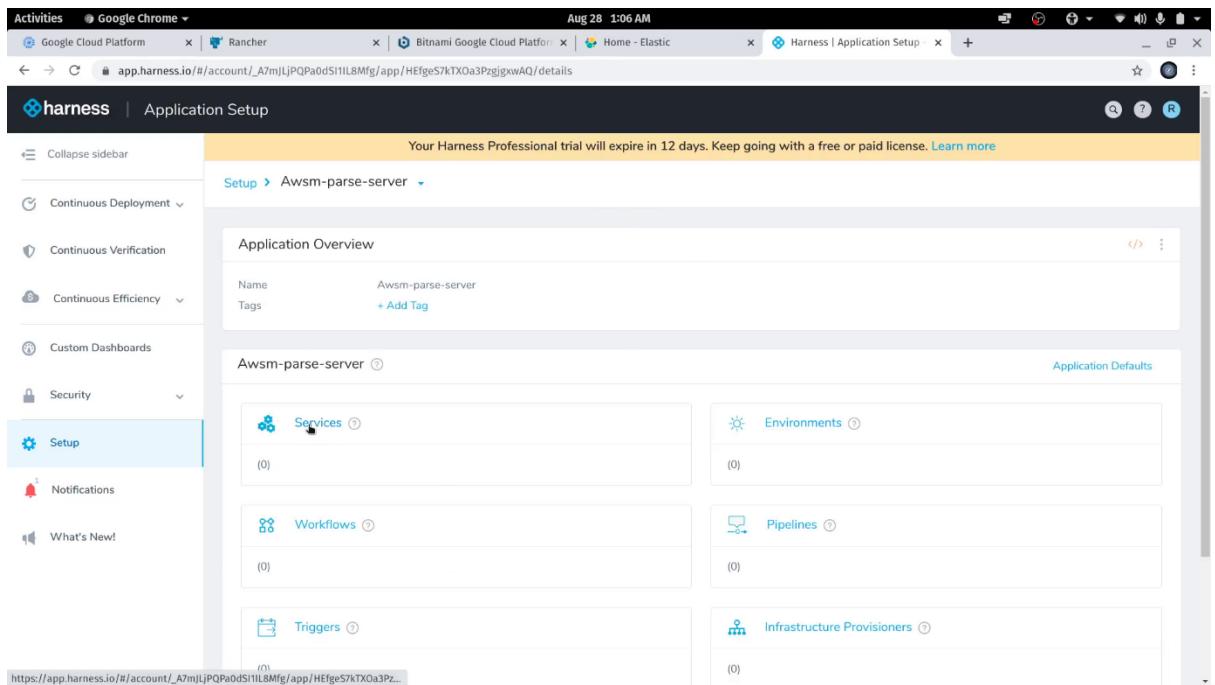
- Now, let's Add Application from Setup.

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 note 'No Results'. 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 **Awsm-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, there's 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, there 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 content area displays the setup for a 'bitnami-parse-server'. It shows the Manifest Format as 'Helm Chart from Helm Repository' and the Helm Repository as 'Bitnami Helm Hub'. The Chart Name is 'bitnami/pars' and the Chart Version is '11.0.4'. Below this, there's a Configuration section with tabs for Config Variables, Config Files, and Values YAML Override, each with 'Add' buttons.

- Now, let's head to Environments.

The screenshot shows the Harness Application Setup interface. The left sidebar has the 'Setup' option selected. The main content area shows the 'Application Overview' for the 'Awm-parse-server' application. It lists the Name as 'Awm-parse-server' and Tags as '+ Add Tag'. Below this, there's a grid of cards for different application components: Services (1 bitnami-parse-server), Environments (0), Workflows (0), Pipelines (0), Triggers (0), and Infrastructure Provisioners (0).

- 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 area says 'No Results' and 'There are no Environments.' with a blue 'Add Environment' button below it.

- 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*' (awsm-parse), 'Description' (empty), and 'Environment Type*' (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 shows the current path: Setup > Awsm-parse-server > Environments > awsm-parse. A green banner at the top right says "Added Successfully". Below it, a message states: "Your Harness Professional trial will expire in 12 days. Keep going with a free or paid license. [Learn more](#)". The "Environment Overview" section shows the environment details: Name: awsm-parse, Environment Type: Production, Tags: + Add Tag. The "Infrastructure Definitions" section lists 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 a message: "No Service Configuration Overrides." with 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 has "Setup" selected. The main area shows the "Awsm-parse-server" environment with various components listed in boxes: Services (1 bitnami-parse-server), Environments (1 awsm-parse), Workflows (0), Pipelines (0), Triggers (0), and Infrastructure Provisioners (0). Below these, there's a section for "Application Resources" 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. 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.

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

The screenshot shows a "Workflow" dialog box over a blurred background of the Harness interface. The dialog has fields for Name (awsm-parse-rolling-deployment), Description (Enter a Description (optional)), Workflow Type (Rolling Deployment), Environment (awsm-parse), Service (bitnami-parse-server), and Infrastructure Definition (awsm-cluster-infra). A "Submit" button at the bottom is highlighted with a mouse cursor.

- 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.' with a 'Learn more' link.

- 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 navigation path is: Setup > Awsm-parse-server. The 'Application Overview' section shows the name 'Awsm-parse-server' and tags '(1) awsm-parse'. Below this, under 'Awsm-parse-server', 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 'Application Defaults' link is visible in the top right of the main content area. A message at the top states: 'Your Harness Professional trial will expire in 12 days. Keep going with a free or paid license.' with a 'Learn more' link.

- Now, let's Add Pipeline.

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

Search + Add Pipeline

No Results

There are no Pipelines.

Add Pipeline

- Enter a name for our Pipeline and Submit.

Add Pipeline

Name*

Description

Submit

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

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

Pipeline Overview

Name: Deploy-awsm-parse
Tags: + Add Tag

Pipeline Stages

Click to add Pipeline Stage

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

Execution Step Approval Step

Ticketing System*

User Group(s)*

Please ensure that all User Groups have Application: Deployments: read permission.

Timeout*

Execute in Parallel with Previous Step

Advanced Settings

Submit

- 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.

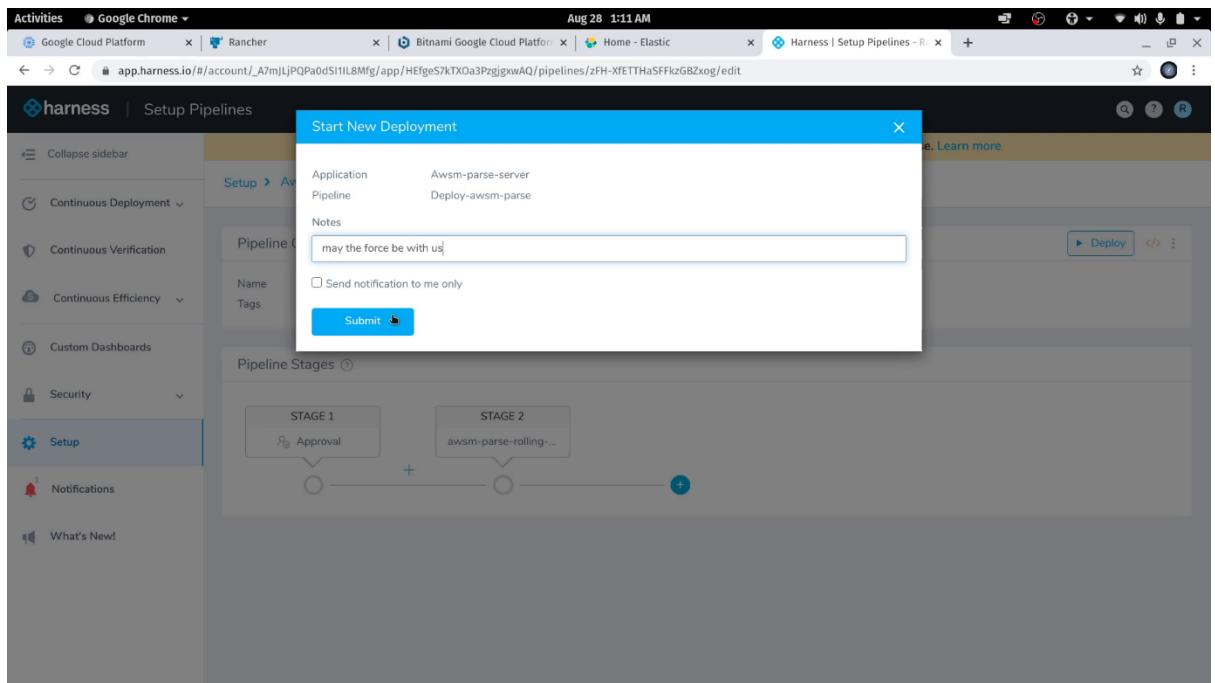
The screenshot shows the Harness web interface for setting up pipelines. On the left, there's a sidebar with various options like Continuous Deployment, Continuous Verification, and Security. The 'Setup' option is currently selected. The main area is titled 'Pipeline Overview' for a pipeline named 'Deploy-awsm-parse'. 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.

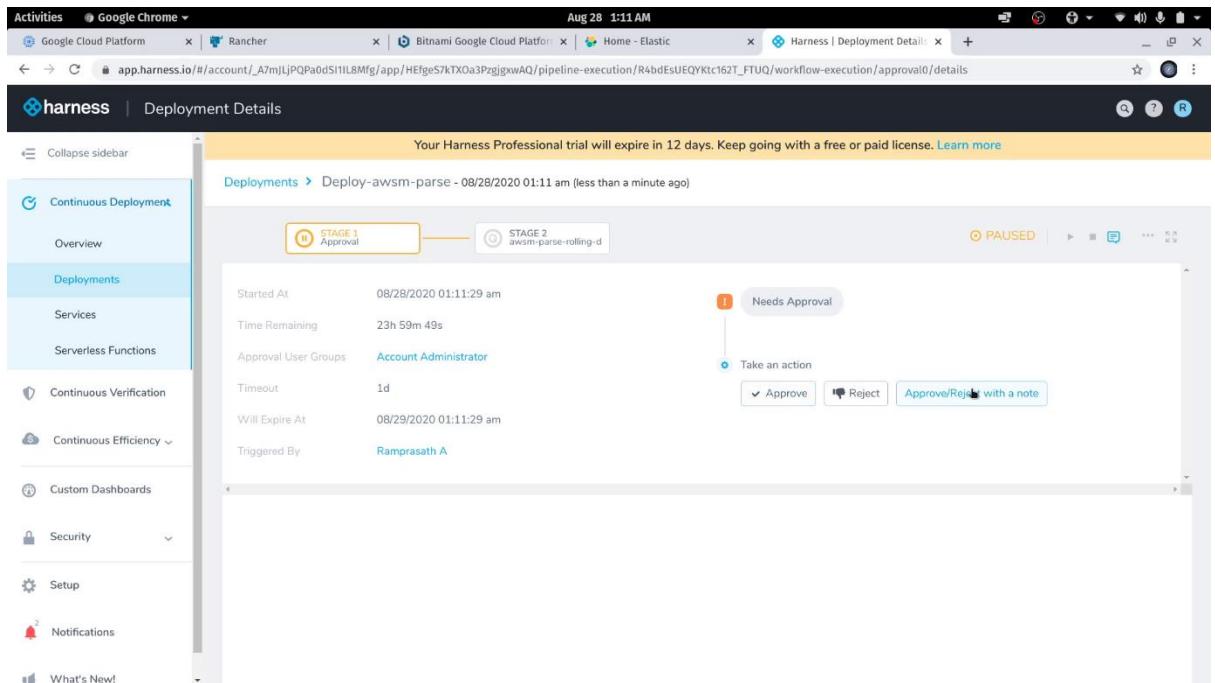
- Click Deploy.

This screenshot is nearly identical to the previous one, showing the Harness pipeline setup interface. The main 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 pipeline structure and stages remain the same as in the previous screenshot.

- 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 visible above the note.

- Deployment has been approved.

The screenshot shows the same Harness Deployment Details page as before, but the deployment status is now 'RUNNING' (indicated by a green circle icon). 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-t). 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 "Rolling" step is currently active. On the right, the "Rollout Deployment" section shows the deployment artifacts: "awsm-parse-server", "bitnami-parse-server", "awsm-parse", and "awsm-parse-rolling-depl...". The "Artifacts Instances Deployed" section lists "None" and "Infrastructure Triggered By" as "None". The "Details" section shows the deployment started at 08/28/2020 01:11:59 am. The "Logs" section displays INFO messages from 2020-08-28 01:12:03, including "Fetching values.yaml from helm chart for Service".

- Deployment successful.

The screenshot shows the Harness Deployment Details page for the same deployment "Deploy-awsm-parse" now successfully completed on Aug 28 1:14 AM (3 mins ago). The status is now "SUCCESS". The deployment artifacts remain the same: "awsm-parse-server", "bitnami-parse-server", "awsm-parse", and "awsm-parse-rolling-depl...". The "Post-Deployment" step has been added to the workflow diagram. The "Logs" section shows the deployment completed successfully with no errors.

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

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

- 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.

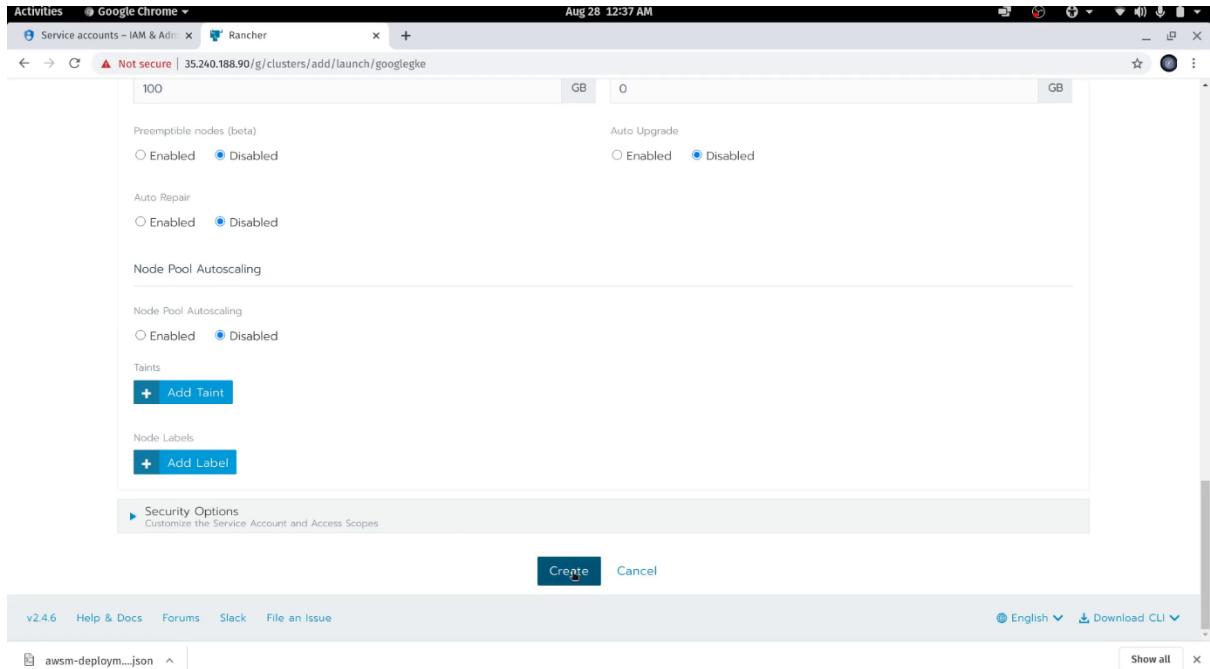
17. Note: Due to extensive time spent on learning new concepts and new tools from scratch, I was not able to setup Velero for backing up data.

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.

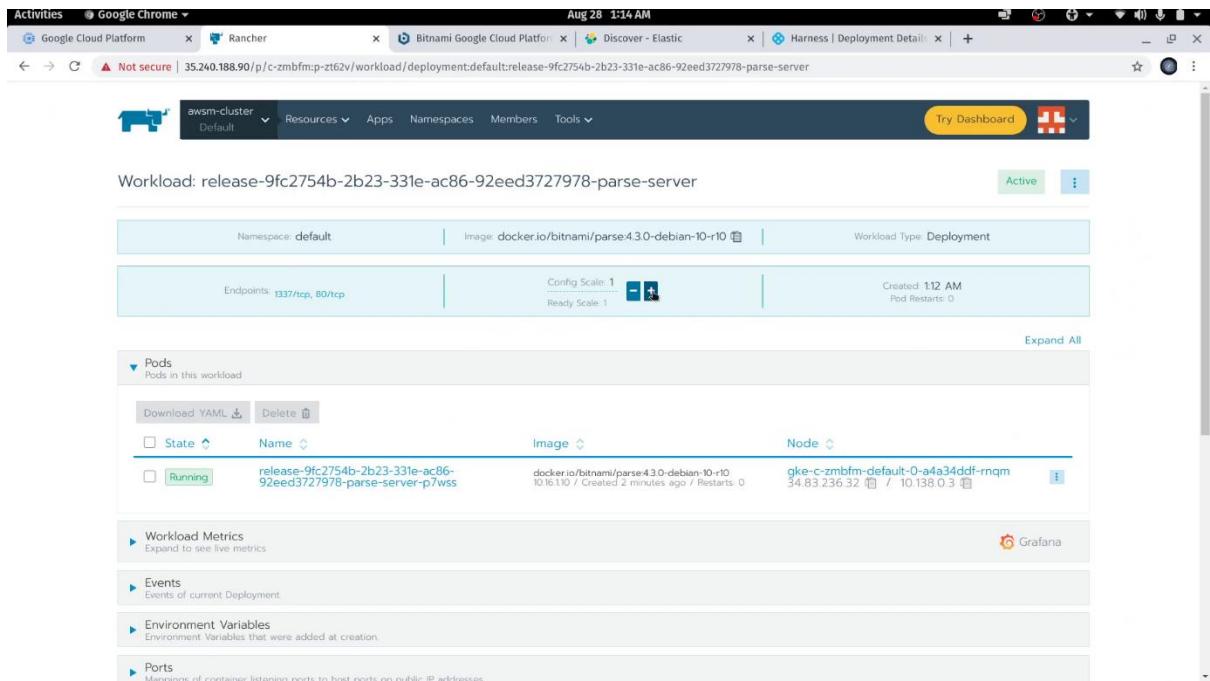
To enable node pool autoscaling, select enable under node pool autoscaling



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

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 UI for a workload named 'release-9fc2754b-2b23-331e-ac86-92eed3727978-parse-server'. It displays the namespace (default), image (docker.io/bitnami/parse:4.3.0-debian-10-r10), and workload type (Deployment). The endpoints listed are 1337/tcp and 80/tcp. A single pod named 'release-9fc2754b-2b23-331e-ac86-92eed3727978-parse-server-p7wss' is shown, which is running on the node 'gke-c-zmbfm-default-0-a4a34ddf-rmqm'. The pod was created at 112 AM and has 0 restarts. The UI also includes sections for 'Pods', 'Workload Metrics' (using Grafana), 'Events', 'Environment Variables', and 'Ports'.

- 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/>