

For my final project, I chose to do Project number 1 for the fictional ABC Technologies company.

Resources can be located at:

<https://github.com/clong1969/PGPDevOps>

<https://hub.docker.com/repository/docker/clong1969/abctech/general>

1. The Requirements:

ABC technologies is a leading online retail store. ABC has recently acquired a large retail offline business store. The business store has large number of stores across the globe but is following conventional pattern of development and deployment. As a result, it has landed to great loss and are facing below challenges.

- low available
- low scalable
- low Performance
- Hard built and maintained.
- Developed and deployed.

is time consuming ABC will acquire the data from all these storage systems and plan to use it for analytics and prediction of the firm's growth and sales prospect. In the first phase ABC has to create the servlets to Add a product and Display product details. Add servlet dependencies required to compile the servlets. Create an HTML page which will be used to add a product. Team is using git to keep all the source code. ABC has decided to use DevOps model and Once source code is available in GitHub, we need to integrate it with Jenkins and provide continuous build generation for continuous Delivery, integrate with Ansible and Kubernetes for deployment. Use docker hub to pull and push images between ansible and Kubernetes.

2. Goals:

- Implement CICD such that ABC Company to is able to be

- Highly available
- Highly scalable
- Highly Performant
- Easily built and maintained
- Developed and deployed quickly

3. Tasks:

We need to develop a CICD pipeline to automate the software development, testing, package, deploy reducing the time to market of app and ensuring good quality service is experienced by end users. In this project we need to

- Push the code to our GitHub repository.
- Create a continuous integration pipeline using Jenkins to compile, test and package the code present in GitHub
- Write docker file to push the war file to tomcat server.

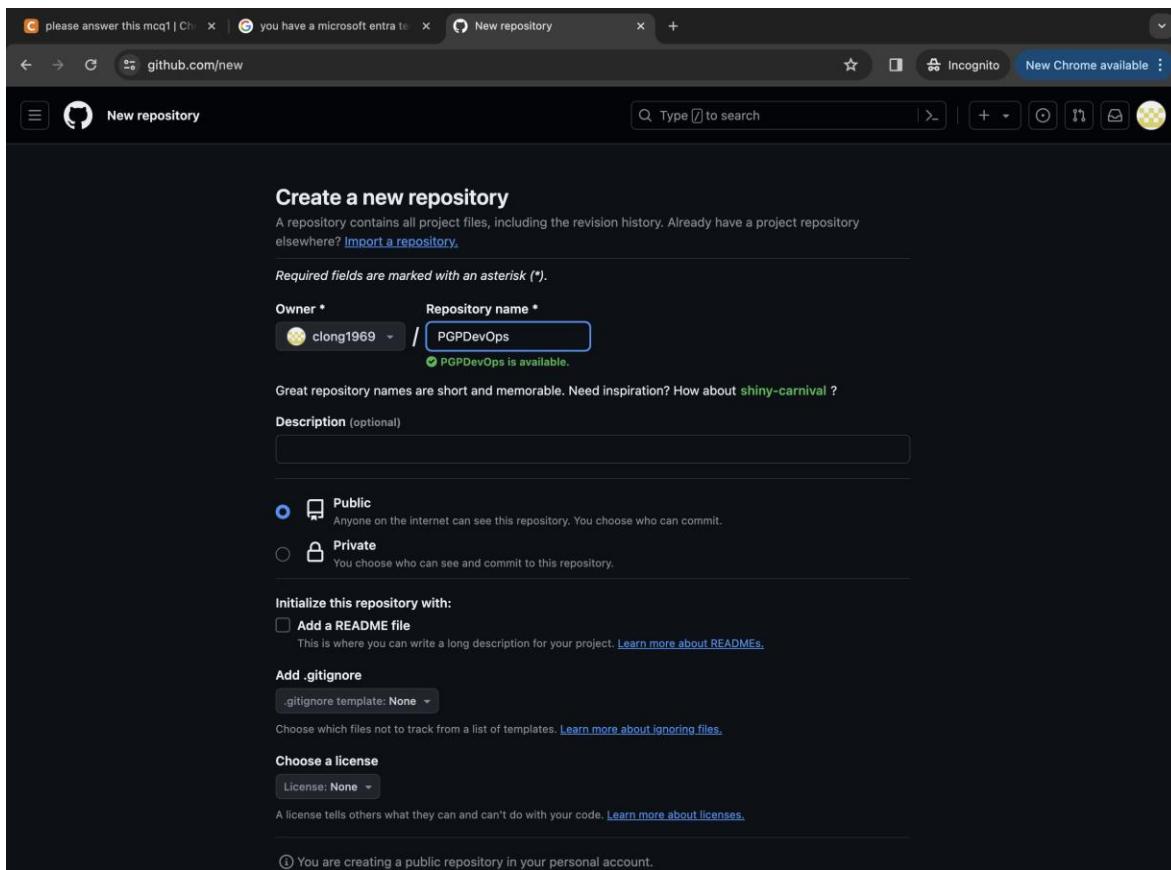
- Integrate docker with Ansible and write playbook5.Deploy artifacts to Kubernetes cluster.
- Monitor resources using Grafana.

Task1: Push supplied code to repository.

The code for this project was supplied by Edureka in their GitHub repository. In order to complete this overall project, I would need to get that code and place it in my repository. To do this, I completed the following steps.

I created a new repository in my GitHub account.

<https://github.com/clong1969/PGPDevOps.git>



Copied the code to my local machine by issuing a git command. I have used git before, so it was already installed on my machine.

I then compiled the code. In order to do this, I installed Maven locally and compiled the source code that was downloaded in the previous step. This was accomplished by running: mvn compile.

```
ABC Technologies -- zsh -- 80x24
-io/2.6/commons-io-2.6.jar (215 kB at 833 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/ow2/asm/asm/9.4/asm-9.4.jar (122 kB at 407 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/com/thoughtworks/qdox/qdox/2.0.3/qdox-2.0.3.jar (334 kB at 995 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/3.5.0/plexus-utils-3.5.0.jar (267 kB at 761 kB/s)
[INFO] Changes detected - recompiling the module! :source
[INFO] Compiling 3 source files with javac [debug target 1.8] to target/classes
[WARNING] bootstrap class path is not set in conjunction with -source 8
    not setting the bootstrap class path may lead to class files that cannot run on JDK 8
        --release 8 is recommended instead of -source 8 -target 1.8 because it sets
        the bootstrap class path automatically
[WARNING] source value 8 is obsolete and will be removed in a future release
[WARNING] target value 8 is obsolete and will be removed in a future release
[WARNING] To suppress warnings about obsolete options, use -Xlint:-options.
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 14.824 s
[INFO] Finished at: 2024-04-26T20:49:56-05:00
[INFO] -----
charlielong@Charlies-Laptop ABC Technologies %
```

Once the code compile was complete, I tested it to make sure it compiled correctly by running: mvn test.

```
ABC Technologies -- zsh -- 80x24
urefire/common-junit4/3.2.2/common-junit4-3.2.2.jar (26 kB at 418 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/surefire/common-junit3/3.2.2/common-junit3-3.2.2.jar (12 kB at 186 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/surefire/common-java5/3.2.2/common-java5-3.2.2.jar (18 kB at 270 kB/s)
[INFO] -----
[INFO] T E S T S
[INFO] -----
[INFO] Running com.abc.dataAccessObject.ProductImpTest
[INFO] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.015 s -
- in com.abc.dataAccessObject.ProductImpTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 7.156 s
[INFO] Finished at: 2024-04-26T20:53:03-05:00
[INFO] -----
charlielong@Charlies-Laptop ABC Technologies %
```

Now that I had a good compile, the code had to be packaged for distribution. This was done by running: mvn package.

```
ABC Technologies — zsh — 80x24
s/plexus-utils/3.1.0/plexus-utils-3.1.0.jar (262 kB at 406 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/com/thoughtworks/x
stream/xstream/1.4.10/xstream-1.4.10.jar (590 kB at 773 kB/s)
[INFO] Packaging webapp
[INFO] Assembling webapp [ABCtechnologies] in [/Users/charlielong/Downloads/Indu
stry Grade Project I - Java Project/ABC Technologies/target/ABCtechnologies-1.0]
[INFO] Processing war project
[INFO] Copying webapp resources [/Users/charlielong/Downloads/Industry Grade Pro
ject I - Java Project/ABC Technologies/src/main/webapp]
[INFO] Webapp assembled in [24 msecs]
[INFO] Building war: /Users/charlielong/Downloads/Industry Grade Project I - Jav
a Project/ABC Technologies/target/ABCtechnologies-1.0.war
[INFO]
[INFO] --- jacoco:0.8.6:report (jacoco-site) @ ABCtechnologies ---
[INFO] Loading execution data file /Users/charlielong/Downloads/Industry Grade P
roject I - Java Project/ABC Technologies/target/jacoco.exec
[INFO] Analyzed bundle 'RetailModule' with 2 classes
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 5.130 s
[INFO] Finished at: 2024-04-26T20:56:01-05:00
[INFO] -----
charlielong@Charlies-Laptop ABC Technologies %
```

Once the code was compiled, I put all the resources into the same folder on my local machine and got it ready to upload to GitHub. This was done by running: git init.

I ran a git add ..

```
ABC Technologies — zsh — 117x32
charlielong@Charlies-Laptop ABC Technologies % git --version
git version 2.39.3 (Apple Git-146)
charlielong@Charlies-Laptop ABC Technologies % ls
README.md      pom.xml      pom.xml.bak    src
charlielong@Charlies-Laptop ABC Technologies % git init
Initialized empty Git repository in /Users/charlielong/Downloads/Industry Grade Project I - Java Project/ABC Technolo
gies/.git/
charlielong@Charlies-Laptop ABC Technologies % git add .
charlielong@Charlies-Laptop ABC Technologies % git status
On branch main

No commits yet

Changes to be committed:
(use "git rm --cached <file>..." to unstage)
  new file: .classpath
  new file: .project
  new file: .settings/org.eclipse.jdt.core.prefs
  new file: .settings/org.eclipse.m2e.core.prefs
  new file: README.md
  new file: pom.xml
  new file: pom.xml.bak
  new file: src/main/java/com/abc/RetailModule.java
  new file: src/main/java/com/abc/dataAccessObject/RetailAccessObject.java
  new file: src/main/java/com/abc/dataAccessObject/RetailDataImp.java
  new file: src/main/webapp/WEB-INF/web.xml
  new file: src/main/webapp/index.jsp
  new file: src/test/java/com/abc/dataAccessObject/ProductImpTest.java

charlielong@Charlies-Laptop ABC Technologies %
```

Once the files were added, I issued git commit -m "first commit" to prepare files for upload.

```
ABC Technologies -- zsh -- 117x32

error: src refspec main does not match any
error: failed to push some refs to 'origin'
[charliealong@Charlies-Laptop ABC Technologies % git commit -m "first commit"
[main (root-commit) 619ae75] first commit
Committer: Charlie Long <charliealong@Charlies-Laptop.attlocal.net>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

git config --global --edit

After doing this, you may fix the identity used for this commit with:

git commit --amend --reset-author

13 files changed, 313 insertions(+)
create mode 100644 .classpath
create mode 100644 .project
create mode 100644 .settings/org.eclipse.jdt.core.prefs
create mode 100644 .settings/org.eclipse.m2e.core.prefs
create mode 100644 README.md
create mode 100644 pom.xml
create mode 100644 pom.xml.bak
create mode 100644 src/main/java/com/abc/RetailModule.java
create mode 100644 src/main/java/com/abc/dataAccessObject/RetailAccessObject.java
create mode 100644 src/main/java/com/abc/dataAccessObject/RetailDataImp.java
create mode 100644 src/main/webapp/WEB-INF/web.xml
create mode 100644 src/main/webapp/index.jsp
create mode 100644 src/test/java/com/abc/dataAccessObject/ProductImpTest.java
charliealong@Charlies-Laptop ABC Technologies %
```

I then tried pushing the code to my GitHub repository. GitHub discontinued support for username and password authentication, so I had to install key on my local machine to provide authentication.

```
ABC Technologies -- zsh -- 117x32

git commit --amend --reset-author

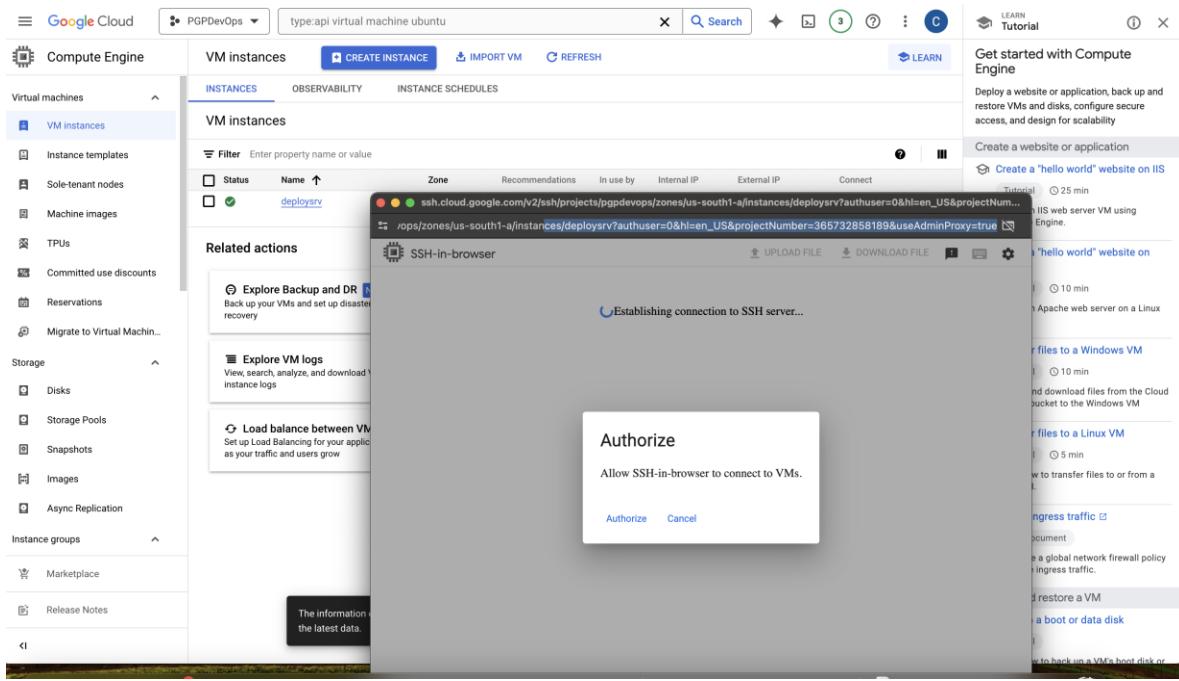
1 file changed, 1 insertion(+)
error: remote origin already exists.
Username for 'https://github.com': clong1969
>Password for 'https://clong1969@github.com':
remote: Support for password authentication was removed on August 13, 2021.
remote: Please see https://docs.github.com/get-started/getting-started-with-git/about-remote-repositories#cloning-with-https-urls for information on currently recommended modes of authentication.
fatal: Authentication failed for 'https://github.com/clong1969/PGPDevOps.git/'
[charliealong@Charlies-Laptop ABC Technologies % git config --global credential.helper store
[charliealong@Charlies-Laptop ABC Technologies % git push -u origin main
Username for 'https://github.com': dfjdjd
>Password for 'https://dfjdjd@github.com':
remote: Support for password authentication was removed on August 13, 2021.
remote: Please see https://docs.github.com/get-started/getting-started-with-git/about-remote-repositories#cloning-with-https-urls for information on currently recommended modes of authentication.
fatal: Authentication failed for 'https://github.com/clong1969/PGPDevOps.git/'
[charliealong@Charlies-Laptop ABC Technologies % git push -u origin main
Username for 'https://github.com': clong1969
>Password for 'https://clong1969@github.com':
Enumerating objects: 32, done.
Counting objects: 100% (32/32), done.
Delta compression using up to 8 threads
Compressing objects: 100% (23/23), done.
Writing objects: 100% (32/32), 4.72 KiB | 4.72 MiB/s, done.
Total 32 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/clong1969/PGPDevOps.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
charliealong@Charlies-Laptop ABC Technologies %
```

Here is my GitHub repo after the initial upload completed. At this time, all that was present were the original source files compiled.

The screenshot shows a GitHub repository page for 'PGPDevOps' owned by 'clong1969'. The repository has 1 branch and 0 tags. The code tab is selected, showing a single commit from 'Charlie Long' and 'Charlie Long' at 6e9f37b, which is 7 minutes ago. The commit message is 'first commit'. The repository contains several files: .settings, src, .classpath, .project, README.md, pom.xml, and pom.xml.bak. The README file contains the text 'abctechnologies code' and '# assignment1'. The repository has 0 stars, 0 forks, and 0 watching. It is categorized under 'PGP Industry Grade Project'. There are sections for Readme, Activity, Releases, Packages, Languages (Java 100.0%), and Suggested workflows (Scala).

Task 2: Create a continuous integration pipeline using Jenkins.

For this task, I created a single vm in Google Cloud and installed Jenkins, Java, Git, Docker and Maven.



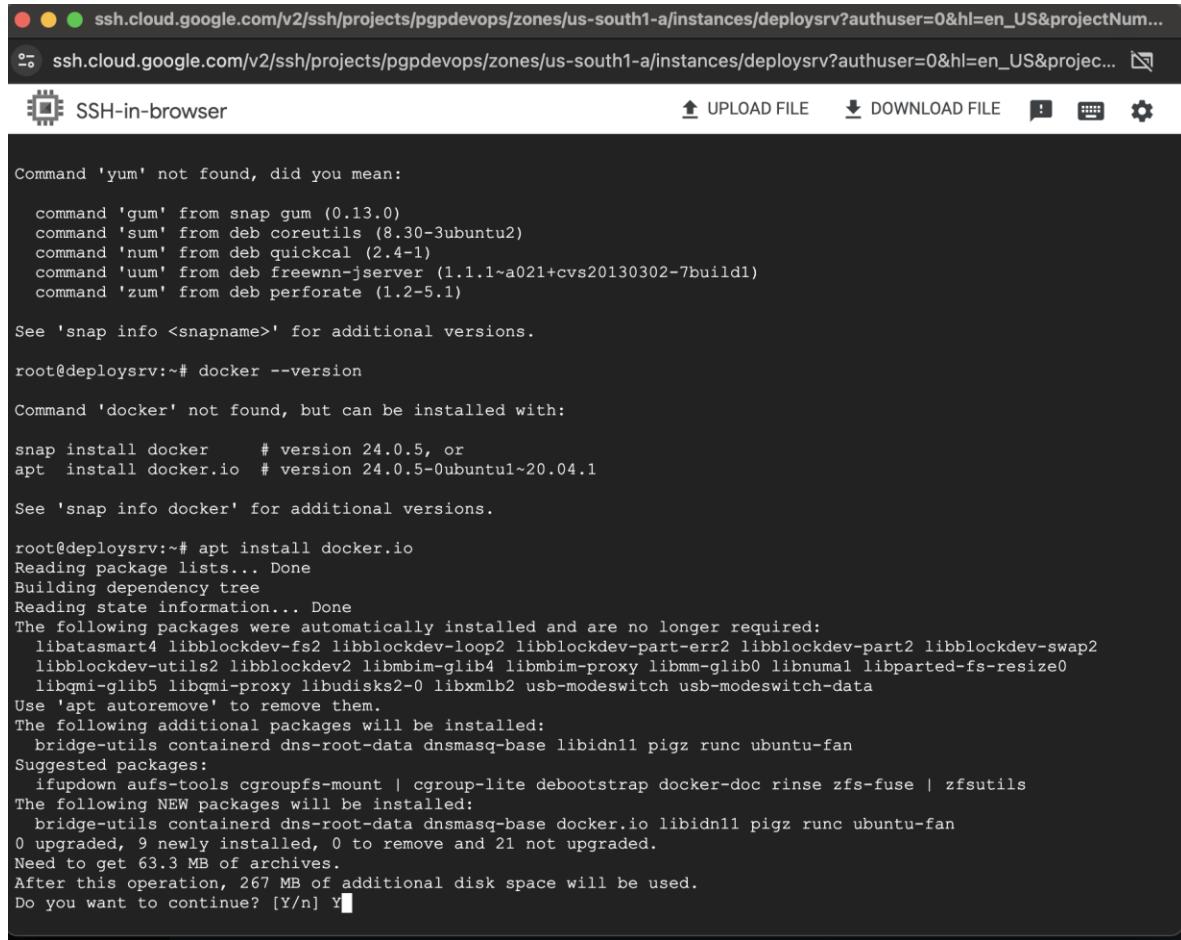
Made SSH connection to newly created VM to install required software.

In order to install Jenkins, I first had to install Java. To do this, I switched to the root user on my Ubuntu VM by typing: sudo su –

Verify git is installed: git –version. VM has version 2.25.1

Install Java: `sudo apt install openjdk-17-jdk -y`

I then installed Docker on the same machine where Jenkins would be installed.



ssh.cloud.google.com/v2/ssh/projects/pgpdevops/zones/us-south1-a/instances/deloysrv?authuser=0&hl=en_US&projectNum...

ssh.cloud.google.com/v2/ssh/projects/pgpdevops/zones/us-south1-a/instances/deloysrv?authuser=0&hl=en_US&projec... 🔍

SSH-in-browser

UPLOAD FILE DOWNLOAD FILE ! ⚙️

```
Command 'yum' not found, did you mean:
  command 'gum' from snap gum (0.13.0)
  command 'sum' from deb coreutils (8.30-3ubuntu2)
  command 'num' from deb quickcal (2.4-1)
  command 'uum' from deb freewnn-jserver (1.1.1~a021+cvs20130302-7build1)
  command 'zum' from deb perforate (1.2-5.1)

See 'snap info <snapname>' for additional versions.

root@deloysrv:~# docker --version

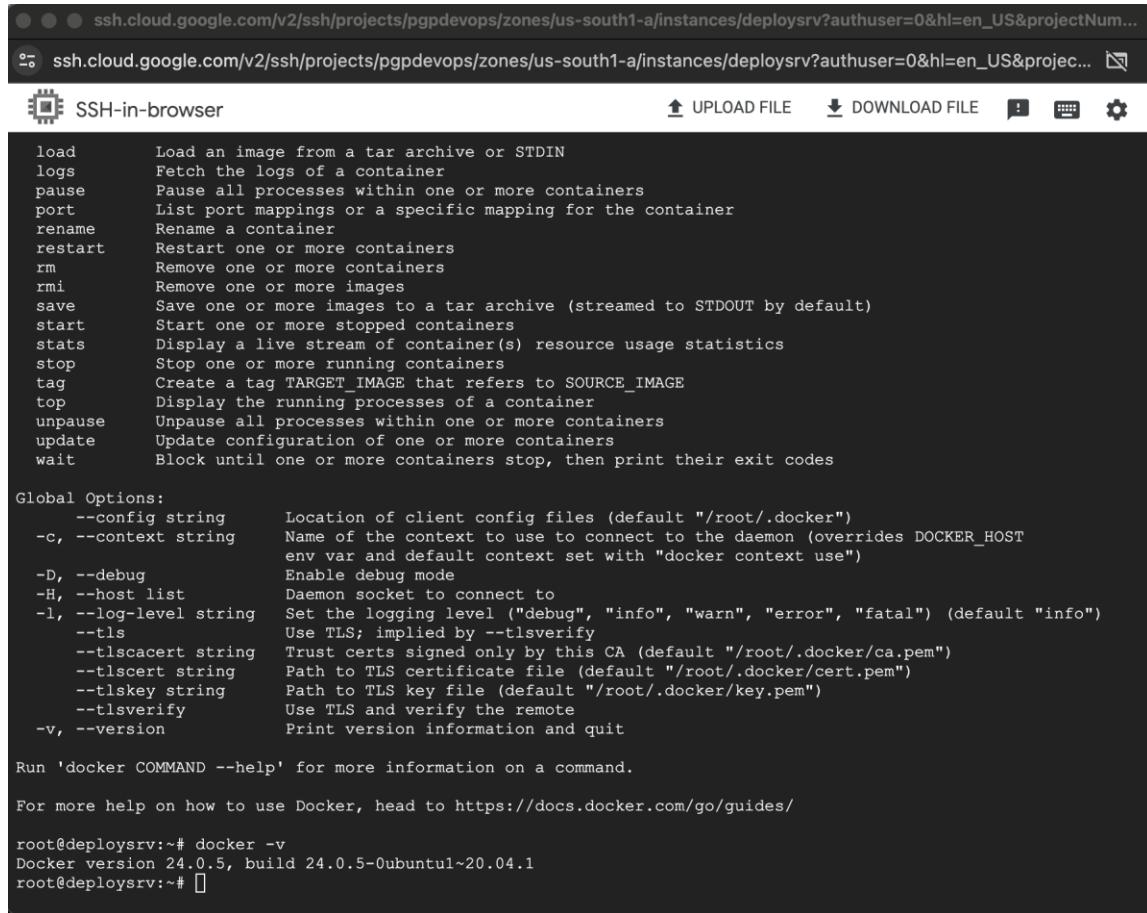
Command 'docker' not found, but can be installed with:

snap install docker      # version 24.0.5, or
apt install docker.io   # version 24.0.5-0ubuntu1~20.04.1

See 'snap info docker' for additional versions.

root@deloysrv:~# apt install docker.io
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libataSMART4 libblkdev-fs2 libblkdev-loop2 libblkdev-part-err2 libblkdev-part2 libblkdev-swap2
  libblkdev-utils2 libblkdev2 libmbim-glib4 libmbim-proxy libmm-glib0 libnumal libparted-fs-resize0
  libqmi-glib5 libqmi-proxy libudisks2-0 libxmlb2 usb-modeswitch usb-modeswitch-data
Use 'apt autoremove' to remove them.
The following additional packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base libidn11 pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-doc rinse zfs-fuse | zfsutils
The following NEW packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base docker.io libidn11 pigz runc ubuntu-fan
0 upgraded, 9 newly installed, 0 to remove and 21 not upgraded.
Need to get 63.3 MB of archives.
After this operation, 267 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
```

Verified that installation was successful by issuing docker -v to check the version.



The screenshot shows a terminal window titled "SSH-in-browser" connected to a Google Cloud instance. The terminal displays the Docker command-line interface's help documentation. The output includes a list of Docker commands with their descriptions, global options, and a note about running 'docker COMMAND --help' for more information. At the bottom, it shows the Docker version and a root prompt.

```
● ● ● ssh.cloud.google.com/v2/ssh/projects/pgpdevops/zones/us-south1-a/instances/deplowsrv?authuser=0&hl=en_US&projectNum...
● ssh.cloud.google.com/v2/ssh/projects/pgpdevops/zones/us-south1-a/instances/deplowsrv?authuser=0&hl=en_US&projec... 🖌

SSH-in-browser
SSH-in-browser

load      Load an image from a tar archive or STDIN
logs      Fetch the logs of a container
pause     Pause all processes within one or more containers
port      List port mappings or a specific mapping for the container
rename   Rename a container
restart  Restart one or more containers
rm       Remove one or more containers
rmi      Remove one or more images
save      Save one or more images to a tar archive (streamed to STDOUT by default)
start    Start one or more stopped containers
stats    Display a live stream of container(s) resource usage statistics
stop     Stop one or more running containers
tag      Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
top      Display the running processes of a container
unpause  Unpause all processes within one or more containers
update   Update configuration of one or more containers
wait    Block until one or more containers stop, then print their exit codes

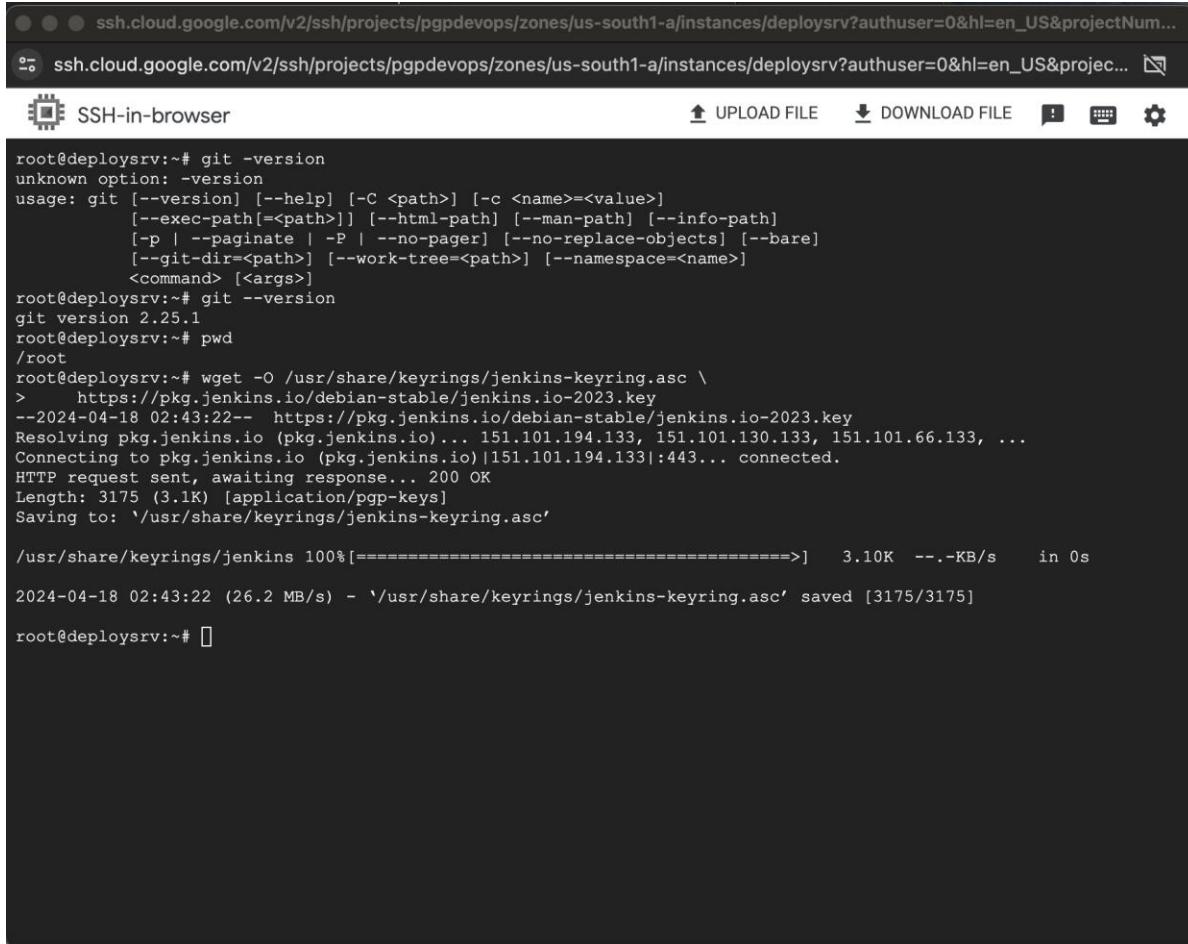
Global Options:
  --config string          Location of client config files (default "/root/.docker")
  -c, --context string     Name of the context to use to connect to the daemon (overrides DOCKER_HOST
                           env var and default context set with "docker context use")
  -D, --debug              Enable debug mode
  -H, --host list           Daemon socket to connect to
  -l, --log-level string   Set the logging level ("debug", "info", "warn", "error", "fatal") (default "info")
  --tlscacert string        Trust certs signed only by this CA (default "/root/.docker/ca.pem")
  --tlscert string          Path to TLS certificate file (default "/root/.docker/cert.pem")
  --tlskey string            Path to TLS key file (default "/root/.docker/key.pem")
  --tlsverify               Use TLS and verify the remote
  -v, --version             Print version information and quit

Run 'docker COMMAND --help' for more information on a command.

For more help on how to use Docker, head to https://docs.docker.com/go/guides/

root@deplowsrv:~# docker -v
Docker version 24.0.5, build 24.0.5-Ubuntu1~20.04.1
root@deplowsrv:~# 
```

I then started an SSH terminal from within Google Cloud and started Jenkins installation.



The screenshot shows a terminal window titled "SSH-in-browser" connected to a Google Cloud VM. The terminal output is as follows:

```
root@deloysrv:~# git -version
unknown option: -version
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]
root@deloysrv:~# git --version
git version 2.25.1
root@deloysrv:~# pwd
/root
root@deloysrv:~# wget -O /usr/share/keyrings/jenkins-keyring.asc \
> https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
--2024-04-18 02:43:22-- https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
Resolving pkg.jenkins.io (pkg.jenkins.io)... 151.101.194.133, 151.101.130.133, 151.101.66.133, ...
Connecting to pkg.jenkins.io (pkg.jenkins.io)|151.101.194.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3175 (3.1K) [application/pgp-keys]
Saving to: '/usr/share/keyrings/jenkins-keyring.asc'

/usr/share/keyrings/jenkins 100%[=====] 3.10K --.-KB/s in 0s
2024-04-18 02:43:22 (26.2 MB/s) - '/usr/share/keyrings/jenkins-keyring.asc' saved [3175/3175]
root@deloysrv:~# 
```

Once the installation was completed, I could Start Jenkins: `systemctl start Jenkins`. Throughout the course of completing this project I repeatedly checked the status by issuing a `systemctl status Jenkins`. There were also some instances where Jenkins froze on me, so I had to restart the application by issuing a `systemctl restart Jenkins` command.

Connect to Jenkins from my local web browser by typing VMs public IP address and port 8080. Enter initial password and go through initial setup. Once done, I was directed to Jenkins dashboard.

The screenshot shows the Jenkins dashboard at the URL 34.174.188.236:8080. The top navigation bar includes links for 'New Chrome available', 'Charlie', and 'log out'. The left sidebar features links for 'New Item', 'People', 'Build History', 'Manage Jenkins', and 'My Views'. Below the sidebar, there are two sections: 'Build Queue' (empty) and 'Build Executor Status' (showing 1 Idle and 2 Idle). The main content area is titled 'Welcome to Jenkins!' with the sub-instruction 'Start building your software project'. It includes a 'Create a job' button, a 'Set up a distributed build' section with 'Set up an agent' and 'Configure a cloud' options, and a link to 'Learn more about distributed builds'. At the bottom right, there are links for 'REST API' and 'Jenkins 2.440.3'.

Throughout the use of Jenkins to complete this project, multiple plugins would be required to complete all the steps. This image shows some of the required plugins. Most notably, Kubernetes plugins.

Not secure 34.174.188.236:8080/manage/pluginManager/installed

Dashboard > Manage Jenkins > Plugins

Plugins

- Updates
- Available plugins
- Installed plugins
- Advanced settings
- Download progress

search installed plugins

JUnit Plugin	1265.v650_14fa_f12f0
Allows JUnit-format test results to be published.	
Report an issue with this plugin	
Kubernetes	4203.v1dd4445b_1cf9
This plugin integrates Jenkins with Kubernetes	
Report an issue with this plugin	
Kubernetes + Pipeline + DevOps Steps	1.6
Report an issue with this plugin	
Kubernetes CLI Plugin	1.12.1
Configure kubectl for Kubernetes	
Report an issue with this plugin	
Kubernetes Client API Plugin	6.10.0-240.v57880ce8b_0b_2
Kubernetes Client API plugin for use by other Jenkins plugins.	
Report an issue with this plugin	
Kubernetes Credentials Plugin	0.11
Common classes for Kubernetes credentials	
Report an issue with this plugin	
Kubernetes Credentials Provider	1.262.v2670ef7ea_0c5
Provides a read only credentials store backed by Kubernetes.	
Report an issue with this plugin	
LDAP Plugin	725.v3ch_b_71fb_1a_ef
Adds LDAP authentication to Jenkins	
Report an issue with this plugin	
Mailer Plugin	472.vf7r289a_4b_420
This plugin allows you to configure email notifications for build results.	
Report an issue with this plugin	
Matrix Authorization Strategy Plugin	3.2.2
Offers matrix-based security authorization strategies (global and per-project).	
Report an issue with this plugin	
Matrix Project Plugin	822.824.v14451b_c0fd42
Multi-configuration (matrix) project type.	
Report an issue with this plugin	
Metrics Plugin	4.2.21-449.v6960d7c54c69
This plugin exposes the Metrics API to Jenkins plugins.	
Report an issue with this plugin	

Before I could start with Jenkins, I needed to set up a Kubernetes environment where I could push out the completed container image. I chose to use GKE in Google Cloud. Google provides a nice service for first-time users that deploys a generic cluster at a lower price.

Google Cloud PGPDevOps Search (/) for resources, docs, products, and more Search

Kubernetes Engine Kubernetes clusters CREATE DEPLOY REFRESH

Learn about Enterprise All Fleets

Resource Management Overview Clusters Workloads Teams Applications Secrets & ConfigMaps Storage Object Browser Marketplace

Clusters

Run your business critical workloads faster, safer, and easier at enterprise scale

GKE Enterprise combines multi-cluster and multi-team operations with fully managed security, governance, and service networking components. Enjoy all the benefits of GKE Standard along compliance policies, and provide application visibility with actionable insights and an application-aware network for resiliency.

When you're ready to scale beyond a single team or cluster, GKE Enterprise delivers an integrated and consistent way to configure, secure, protect, and monitor container workloads.

LEARN AND ENABLE

OVERVIEW OBSERVABILITY COST OPTIMIZATION

Filter Enter property name or value

Status	Name ↑	Location	Number of nodes	Total vCPUs	Total memory	Notifications	Labels
<input type="checkbox"/>	my-first-cluster-1	us-central1-c	3	3	5.1 GB	—	⋮

Nodes in the Kubernetes cluster.

The screenshot shows the Google Cloud Platform (GCP) Kubernetes Engine interface. On the left, there's a sidebar with 'Resource Management' sections like Overview, Clusters, Workloads, Teams, Applications, Secrets & ConfigMaps, Storage, Object Browser, and Marketplace. The main area is titled 'Clusters' and shows 'my-first-cluster-1'. Below it, tabs include DETAILS, NODES (which is selected), STORAGE, OBSERVABILITY, LOGS, and APP ERRORS (2). Under 'Node Pools', there's a table with columns: Name, Status, Version, Number of nodes, Machine type, Image type, Autoscaling, and Default IPv4 Pod IP address range. One row is shown: default-pool (Status: Ok, Version: 1.29.3-gke.1093000, 3 nodes, g1-small, Container-Optimized OS with containerd (cos_containerd), Off, 10.28.0.0/14). Under 'Nodes', there's another table with columns: Name, Status, CPU requested, CPU allocatable, Memory requested, Memory allocatable, Storage requested, and Storage allocatable. Three nodes are listed: gke-my-first-cluster-1-default-pool-fd1ec955-g58w (Status: Ready), gke-my-first-cluster-1-default-pool-fd1ec955-sh4v (Status: Ready), and gke-my-first-cluster-1-default-pool-fd1ec955-wbcn (Status: Ready).

My first run through the Jenkins process consisted of running a series of Freestyle Projects. The first objective was to clone and compile my repo to the local Jenkins VM.

The screenshot shows a Jenkins job named 'CloneRepo'. At the top, it says 'CloneRepo [Jenkins]'. The URL is 'Not Secure 34.174.188.236:8080/job/CloneRepo/'. Below that is the Jenkins logo and the word 'Jenkins'. The navigation bar shows 'Dashboard > CloneRepo >'. The main content area has a 'CloneRepo' title and a sidebar with options: Status, Changes, Workspace, Build Now, Configure, Delete Project, and Rename. The 'Build History' section shows a table with a single row for build #1, dated April 19, 2024, at 2:29 AM. It includes links for 'Atom feed for all' and 'Atom feed for failures'.

Dashboard > CloneRepo > Configuration

Configure

Throttle builds ?

Execute concurrent builds if necessary ?

Restrict where this project can be run ?

General

Source Code Management

Build Triggers

Build Environment

Build Steps

Post-build Actions

Advanced ▾

Source Code Management

None

Git ?

Repositories ?

Repository URL ?
https://github.com/clong1969/PGPDevOps.git

Credentials ?
- none -

+ Add ▾

Advanced ▾

This screenshot shows the configuration interface for a Jenkins job named 'CloneRepo'. The 'General' tab is active. Under 'Source Code Management', 'Git' is selected. The 'Repository URL' is set to 'https://github.com/clong1969/PGPDevOps.git'. No credentials are present.

The screenshot shows the Jenkins 'Tools [Jenkins]' interface at the top, with the URL '34.174.188.236:8080/manage/configureTools/' in the address bar. Below the header, the breadcrumb navigation shows 'Dashboard > Manage Jenkins > Tools'. The main content area is titled 'Maven installations'. A 'Add Maven' button is visible. The configuration form for a new Maven installation is displayed, with a 'Name' field containing 'mymaven' (marked as required), an 'Install automatically' checkbox checked, and an 'Install from Apache' section showing 'Version' set to '3.9.6'. At the bottom of the form are 'Save' and 'Apply' buttons.

Maven installations

Add Maven

Maven

Name

mymaven

! Required

Install automatically ?

Install from Apache

Version

3.9.6

Add Installer ▾

Add Maven

Save

Apply

To compile the code, I invoked the TOP-LEVEL MAVEN target from plugins and added the goal of compile.

Dashboard > CloneRepo > Configuration

Delete workspace before build starts

Configure

- General
- Source Code Management
- Build Triggers
- Build Environment**
- Build Steps
- Post-build Actions

Use secret text(s) or file(s) ?

Add timestamps to the Console Output

Configure Kubernetes CLI (kubectl) (deprecated, use the multi credentials one instead) ?

Configure Kubernetes CLI (kubectl) with multiple credentials

Google Cloud Ephemeral Deployer

Inspect build log for published build scans

Setup Kubernetes CLI (kubectl) ?

Terminate a build if it's stuck

Terraform

With Ant ?

Build Steps

Invoke top-level Maven targets ?

Maven Version

mymaven

Goals

compile

Advanced ▾

I entered a cron job in the BUILD TRIGGER so that the job would check the GitHub repo and start running if new files were added.

Dashboard > CloneRepo > Configuration

Configure

- General
- Source Code Management
- Build Triggers**
- Build Environment
- Build Steps
- Post-build Actions

Additional Behaviours

Add ▾

Build Triggers

Trigger builds remotely (e.g., from scripts) ?

Build after other projects are built ?

Build periodically ?

GitHub hook trigger for GITScm polling ?

Poll SCM ?

Schedule ?

⚠ Do you really mean "every minute" when you say "***"? Perhaps you meant "H * * * *" to poll once per hour**
Would last have run at Thursday, May 2, 2024 at 12:55:38 PM Coordinated Universal Time; would next run at Thursday, May 2, 2024 at 12:55:38 PM Coordinated Universal Time.

Ignore post-commit hooks ?

Build Environment

Delete workspace before build starts



Jenkins

Dashboard > CloneRepo > #7 > Console Output

- [Status](#)
- [Changes](#)
- [Console Output](#) Console Output
- [View as plain text](#)
- [Edit Build Information](#)
- [Delete build '#7'](#)
- [Timings](#)
- [Git Build Data](#)
- [Previous Build](#)

Console Output

```

Started by user Charlie
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/CloneRepo
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/CloneRepo/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/clong1969/PGPDevOps.git # timeout=10
Fetching upstream changes from https://github.com/clong1969/PGPDevOps.git
> git --version # timeout=10
> git --version # 'git version 2.25.1'
> git fetch --tags --force --progress -- https://github.com/clong1969/PGPDevOps.git +refs/heads/*:refs/remotes/o
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision f5948ea488185cc2822e893e5004f30d74b33acd (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f f5948ea488185cc2822e893e5004f30d74b33acd # timeout=10
Commit message: "playbook"
> git rev-list --no-walk f5948ea488185cc2822e893e5004f30d74b33acd # timeout=10
[CloneRepo] $ /var/lib/jenkins/tools/hudson.tasks.Maven_MavenInstallation/mymaven/bin/mvn compile
[INFO] Scanning for projects...
[INFO]

```

The next step was a FREESTYLE project to test the compiled code this was triggered by adding a project to watch of the CloneRepo project.

Dashboard > TestRepo > Configuration

- [Configure](#)
- [General](#)
- [Source Code Management](#)
- [Build Triggers](#) Build Triggers
- [Build Environment](#)
- [Build Steps](#)
- [Post-build Actions](#)

Build Triggers

Trigger builds remotely (e.g., from scripts) ?

Build after other projects are built ?

Projects to watch
CloneRepo

Trigger only if build is stable

Trigger even if the build is unstable

Trigger even if the build fails

Always trigger, even if the build is aborted

Build periodically ?

GitHub hook trigger for GITScm polling ?

Poll SCM ?

Build Environment

Delete workspace before build starts

Use secret text(s) or file(s) ?

By adding the TOP-LEVEL Maven target and adding a goal to test, the project would test the compiled code.

Dashboard > TestRepo > Configuration

Configure

- General
- Source Code Management
- Build Triggers
- Build Environment**
- Build Steps
- Post-build Actions

Advanced workspace build steps

- Use secret text(s) or file(s) ?
- Add timestamps to the Console Output
- Configure Kubernetes CLI (kubectl) (deprecated, use the multi credentials one instead) ?
- Configure Kubernetes CLI (kubectl) with multiple credentials
- Google Cloud Ephemeral Deployer
- Inspect build log for published build scans
- Setup Kubernetes CLI (kubectl) ?
- Terminate a build if it's stuck
- Terraform
- With Ant ?

Build Steps

≡ Invoke top-level Maven targets ?

Maven Version
mymaven

Goals
test

Advanced ▾

Dashboard > TestRepo > #6 > Console Output

Console Output

- Status
- </> Changes
- Console Output**
- View as plain text
- Edit Build Information
- Delete build '#6'
- Timings
- Git Build Data
- ← Previous Build

```

Started by upstream project "CloneRepo" build number 7
originally caused by:
  Started by user Charlie
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/TestRepo
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/TestRepo/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/clong1969/PGPDevOps.git # timeout=10
Fetching upstream changes from https://github.com/clong1969/PGPDevOps.git
> git --version # timeout=10
> git --version # 'git version 2.25.1'
> git fetch --tags --force --progress -- https://github.com/clong1969/PGPDevOps.git +refs/heads/
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision f5948ea488185cc2822e893e5004f30d74b33acd (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f f5948ea488185cc2822e893e5004f30d74b33acd # timeout=10
Commit message: "playbook"
> git rev-list --no-walk f5948ea488185cc2822e893e5004f30d74b33acd # timeout=10

```

The next step was to package the code. This would be initiated after the TestRepo project completed.

Dashboard > PackageRepo > Configuration

Configure

- General
- Source Code Management
- Build Triggers
- Build Environment
- Build Steps
- Post-build Actions

Build Triggers

Trigger builds remotely (e.g., from scripts) ?

Build after other projects are built ?

Projects to watch

TestRepo

Trigger only if build is stable

Trigger even if the build is unstable

Trigger even if the build fails

Always trigger, even if the build is aborted

Build periodically ?

GitHub hook trigger for GITScm polling ?

Poll SCM ?

Build Environment

- Delete workspace before build starts
- Use secret text(s) or file(s) ?
- Add timestamps to the Console Output
- Configure Kubernetes CLI (kubectl) (deprecated, use the multi credentials one instead) ?
- Configure Kubernetes CLI (kubectl) with multiple credentials
- Google Cloud Ephemeral Deployer
- Inspect build log for published build scans

The project would package the code due to adding the BUILD STEP, INVOKE TOP-LEVEL MAVEN targets and adding the goal of package.

Dashboard > PackageRepo > Configuration

Configure

- General
- Source Code Management
- Build Triggers
- Build Environment
- Build Steps
- Post-build Actions

Build Environment

Use secret text(s) or file(s) ?

Add timestamps to the Console Output

Configure Kubernetes CLI (kubectl) (deprecated, use the multi credentials one instead) ?

Configure Kubernetes CLI (kubectl) with multiple credentials

Google Cloud Ephemeral Deployer

Inspect build log for published build scans

Setup Kubernetes CLI (kubectl) ?

Terminate a build if it's stuck

Terraform

With Ant ?

Build Steps

Invoke top-level Maven targets ?

Maven Version

mymaven

Goals

package

Advanced ▾

Status
 Changes
 Console Output
 View as plain text
 Edit Build Information
 Delete build '#4'
 Timings
 Git Build Data
 ← Previous Build

Console Output

```

Started by upstream project "TestRepo" build number 6
originally caused by:
  Started by upstream project "CloneRepo" build number 7
originally caused by:
  Started by user Charlie
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/PackageRepo
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/PackageRepo/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/clong1969/PGPDevOps.git # timeout=10
Fetching upstream changes from https://github.com/clong1969/PGPDevOps.git
> git --version # timeout=10
> git --version # 'git version 2.25.1'
> git fetch --tags --force --progress -- https://github.com/clong1969/PGPDevOps.git +ref
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision f5948ea488185cc2822e893e5004f30d74b33acd (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f f5948ea488185cc2822e893e5004f30d74b33acd # timeout=10
Commit message: "playbook"
> git rev-list --no-walk f5948ea488185cc2822e893e5004f30d74b33acd # timeout=10

```

Jenkins

Dashboard > CloneRepo > #1 > Console Output

Status
 Changes
 Console Output
 View as plain text
 Edit Build Information
 Delete build '#1'
 Git Build Data

Console Output

```

Started by user Charlie
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/CloneRepo
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/clong1969/PGPDevOps.git
> git init /var/lib/jenkins/workspace/CloneRepo # timeout=10
Fetching upstream changes from https://github.com/clong1969/PGPDevOps.git
> git --version # timeout=10
> git --version # 'git version 2.25.1'
> git fetch --tags --force --progress -- https://github.com/clong1969/PGPDevOps.git
+refs/heads/*:refs/remotes/origin/* # timeout=10
> git config remote.origin.url https://github.com/clong1969/PGPDevOps.git # timeout=10
> git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 6e9f37bda9b46deb55692a30b8e6373e21ac851a (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f 6e9f37bda9b46deb55692a30b8e6373e21ac851a # timeout=10
Commit message: "first commit"
First time build. Skipping changelog.
Finished: SUCCESS

```

Steps 3, 4 and 5: Write docker file to push war to tomcat server, integrate docker with Ansible and deploy to Kubernetes.

The final step would be to create a container that would include the application and be deployed to container orchestration solution. For this, I included a dockerfile that had the path to the source code, included a web server image and an ansible playbook.

The screenshot shows the Jenkins dashboard with a list of build jobs. The columns include Status, Last Success, Last Failure, and Last Duration. The jobs listed are:

Name	Last Success	Last Failure	Last Duration
CKCD-2	1 day 1 hr #7	1 day 1 hr #6	27 sec
PIPELINE	4 days 1 hr #56	1 day 19 hr #119	30 sec
CloneRepo	1 hr 29 min #7	N/A	4.5 sec
Deploy	1 hr 29 min #7	1 day 1 hr #6	0.39 sec
PackageRepo	1 hr 29 min #6	7 days 19 hr #1	7.9 sec
PGPDevOpsFinal	1 day 2 hr #84	1 day 2 hr #82	30 sec
TestRepo	1 hr 29 min #6	7 days 20 hr #2	6.2 sec

Used the INVOKE ANSIBLE PLAYBOOK and included playbook path
<https://github.com/clong1969/PGPDevOps/ansible-2.yml>

The screenshot shows the configuration page for the 'Deploy' job. Under the 'Build Steps' section, there is a configuration for 'Invoke Ansible Playbook'. The 'Playbook path' field contains the URL: <https://github.com/clong1969/PGPDevOps/ansible-2.yml>. The 'Inventory' section has the radio button 'Do not specify Inventory' selected. There is also a 'Host subset' field at the bottom.

The screenshot shows the Jenkins interface. In the top navigation bar, it says 'Not secure' and the URL '34.174.188.236:8080/job/Deploy/11/console'. Below the header, there's a Jenkins logo and the word 'Jenkins'. The main content area has a sidebar on the left with links like 'Status', 'Changes', 'Console Output' (which is selected), 'View as plain text', 'Edit Build Information', 'Delete build #11', 'Timings', 'Git Build Data', and 'Previous Build'. The right side shows the 'Console Output' tab with a green checkmark icon. The output itself starts with 'Started by user Charlie' and continues with a detailed log of the git fetch and checkout process from a GitHub repository.

```

Started by user Charlie
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/Deploy
The recommended git tool is: NONE
using credential github
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/Deploy/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/clong1969/PGPDevOps.git # timeout=10
Fetching upstream changes from https://github.com/clong1969/PGPDevOps.git
> git --version # timeout=10
> git --version # 'git' version 2.25.1'
using GIT_ASKPASS to set credentials github
> git fetch --tags --force --progress .. https://github.com/clong1969/PGPDevOps.git +refs/heads/*:refs/remotes/origin/*
Checking out Revision 049a1cd376c1d4efde3ad2adae89ade3ba9675fd (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f 049a1cd376c1d4efde3ad2adae89ade3ba9675fd # timeout=10
Commit message: "Initial commit"

```

After the projects were configured, I simply clicked the BUILD NOW icon on the CloneRepo project, and the completion of each project would trigger the next and the whole process completed.

I tested the image by running the docker container locally on the Jenkins server.

Running nodes and pods with workload:

```

charlielong1969@cloudshell:~ (pgpdevops)$ kubectl get nodes
NAME                               STATUS   ROLES      AGE     VERSION
gke-my-first-cluster-1-default-pool-fd1ec955-cczy   Ready    <none>    28h    v1.29.3-gke.1282000
gke-my-first-cluster-1-default-pool-fd1ec955-jyi8   Ready    <none>    28h    v1.29.3-gke.1282000
gke-my-first-cluster-1-default-pool-fd1ec955-rj0z   Ready    <none>    28h    v1.29.3-gke.1282000
charlielong1969@cloudshell:~ (pgpdevops)$ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
abctech-a-577fbf444b-ms7wm   1/1     Running   0          78s
abctech-a-577fbf444b-rdt27   1/1     Running   0          78s
abctech-a-577fbf444b-sn52v   1/1     Running   0          78s

```



Dockerfile consisted of these few lines:

```
FROM tomcat:9  
ADD ABCtechnologies-1.0.war /usr/local/tomcat/webapps  
CMD ["catalina.sh", "run"]  
EXPOSE 8080
```

This process was run numerous times, but the attached image shows that the image was successfully deployed to dockerhub on each successful run. All images from runs are located here:

<https://hub.docker.com/repository/docker/clong1969/abctech/general>

The screenshot shows the Docker Hub interface for the repository `clong1969/abctech`. The repository was last updated 1 day ago. It includes a deployment section and a link to web servers. The Docker commands section shows the command to push a new tag: `docker push clong1969/abctech:tagname`. The Tags section lists 16 tags, with the most recent being 84, pushed a day ago. The Automated Builds section indicates that manual pushing is available and offers an upgrade option.

Tag	OS	Type	Pulled	Pushed
84		Image	16 hours ago	a day ago
83		Image	a day ago	a day ago
80		Image	a day ago	a day ago
79		Image	14 hours ago	a day ago
78		Image	17 hours ago	a day ago

While this process was a good solution for this type of project, in a production environment something more robust would be better suited. For the next attempt, I did a complete pipeline by going into Jenkins, selecting NEW ITEM and selecting PIPELINE.

This would be a single pipeline with a series of stages that would accomplish the same process as the multiple projects above. The process would provide a graphical representation of where the build was at each stage.

Jenkins

Dashboard > PGPDevOpsFinal >

PGPDevOpsFinal

- Status
- Changes
- Build Now
- Configure
- Delete Pipeline
- Full Stage View
- Rename
- Pipeline Syntax

Stage View

Declarative: Tool Install	CloneRepo	Compile	Test	Package	Build Image	Deploy Image	Deploying to Kubernetes	
Average stage times: (Average full run time: ~32s)	105ms	517ms	4s	6s	8s	2s	7s	1s
#84 Apr 30 14:15 commit	106ms	556ms	4s	6s	8s	1s	6s	1s
#85 Apr 30 14:03 No Changes	105ms	487ms	4s	6s	8s	2s	6s	1s

Build History trend ▾

Filter... /

#84 | Apr 30, 2024, 7:15 PM

Configuration

Configure

- Build periodically
- Github hook trigger for GITScm polling
- Poll SCM
- Quiet period
- Trigger builds remotely (e.g., from scripts)

General

Advanced Project Options

Pipeline

Advanced Project Options

Pipeline

Definition

Pipeline script

```

1* pipeline{
2  environment{
3    registry = "clong1969/abctech"
4    registryCredential = "dockerhub"
5  }
6  tools{
7    // what tool version to use for build stages
8    maven "mymaven"
9  }
10  agent any
11  stages{
12    stage ('CloneRepo')
13    {
14      steps{
15        echo "Hello Jenkins"
16      }
17    }
18  }
19}

```

The script is already approved

Use Groovy Sandbox

The screenshot shows the Jenkins interface for a pipeline job named 'PGPDevOpsFinal'. The 'Console Output' tab is selected, displaying the build logs. The logs show a declarative pipeline script being executed, starting with 'Started by user charlie' and performing various steps like cloning a repository and fetching upstream changes.

```

Started by user charlie
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/PGPDevOpsFinal
[Pipeline] {
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Tool Install)
[Pipeline] tool
[Pipeline] envvarsForTool
[Pipeline] }
[Pipeline] // stage
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (CloneRepo)
[Pipeline] tool
[Pipeline] envvarsForTool
[Pipeline] withEnv
[Pipeline] {
[Pipeline] git
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/PGPDevOpsFinal/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/clong1969/PGPDevOps.git # timeout=10
Fetching upstream changes from https://github.com/clong1969/PGPDevOps.git
> git --version # timeout=10
> git --version # git version 2.25.1
> git fetch --tags --force --progress -- https://github.com/clong1969/PGPDevOps.git +refs/heads/*:refs/remotes/origin/* # timeout=10

```

The entire deployment required less than 80 lines of code.

```

pipeline{
    environment {
        registry = "clong1969/abctech"
        registryCredential = 'dockerhub'
    }
    tools{
        // what tool version to use for build stages
        maven 'mymaven'
    }
    agent any
    stages{
        stage ('CloneRepo')
        {
            steps{
                git (branch: 'main', url:'https://github.com/clong1969/PGPDevOps.git')
            }
        }
    }
}

```

```
        }

    }

    stage ('Compile')
    {
        steps{
            sh 'mvn compile'
        }
    }

    stage ('Test')
    {
        steps{
            sh 'mvn test'
        }
    }

    post{
        success{
            junit 'target/surefire-reports/*.xml'
        }
    }
}

stage('Package')
{
    steps{
        sh 'mvn package'
    }
}

stage('Build Image')
{
    steps{
        sh 'cp /var/lib/jenkins/workspace/PGPDevOpsFinal/target/ABCtechnologies-1.0.war .'
    }
}
```

```

script {
    dockerImage = docker.build registry + ":$BUILD_NUMBER"
}
}

stage('Deploy Image')
{
    steps{
        script {
            dockerImage = docker.build registry + ":$BUILD_NUMBER"
            docker.withRegistry( "", registryCredential ) {
                dockerImage.push()
            }
        }
    }
}

stage('Deploying to Kubernetes')
{
    steps{
        sh("sed -ie s/abctech:latest/dockerImage/g deployment.yml")
    }
}

step([$class: 'KubernetesEngineBuilder', projectId: 'pgpdevops', clusterName: 'my-first-cluster-1', location: 'us-central1-c', manifestPattern: 'deployment.yml', credentialsId: 'pgpjenkins'])

}
}

}

```

Once the deployment was complete, I had workloads running in Google Cloud GKE.

Kubernetes Engine

Workloads

REFRESH DEPLOY CREATE JOB DELETE

Cluster Namespace RESET SAVE Saved view

OVERVIEW OBSERVABILITY COST OPTIMIZATION

Filter Is system object : False Filter workloads

Name	Status	Type	Pods	Namespace	Cluster
abctech-pgp	OK	Deployment	3/3	default	my-first-cluster-1
nginx-1	OK	Deployment	3/3	default	my-first-cluster-1
pgp-abctech	Does not have minimum availability	Deployment	0/3	pgpdevops	pgp-abctech-cluster

Tested workload by connecting to application:

Welcome to ABC technologies

This is retail portal

Add Product View Product

Step 6 Monitor resources with Grafana.

This should really be two steps. Prometheus monitors your Kubernetes cluster and Grafana takes the data from the log collections and puts it into an easy to read graphical format which can be customized to the end-user's needs.

For this step, I selected the Prometheus managed service that is included within Google Cloud. This was fairly straightforward process that included selecting the service, deploying the service and selecting the resources that would be monitored. For resources that required an agent, the service deployed agents to selected devices.

This is the Google Cloud interface for monitoring with Prometheus.

Google Cloud PGPDevOps

Observability Monitoring GKE Clusters

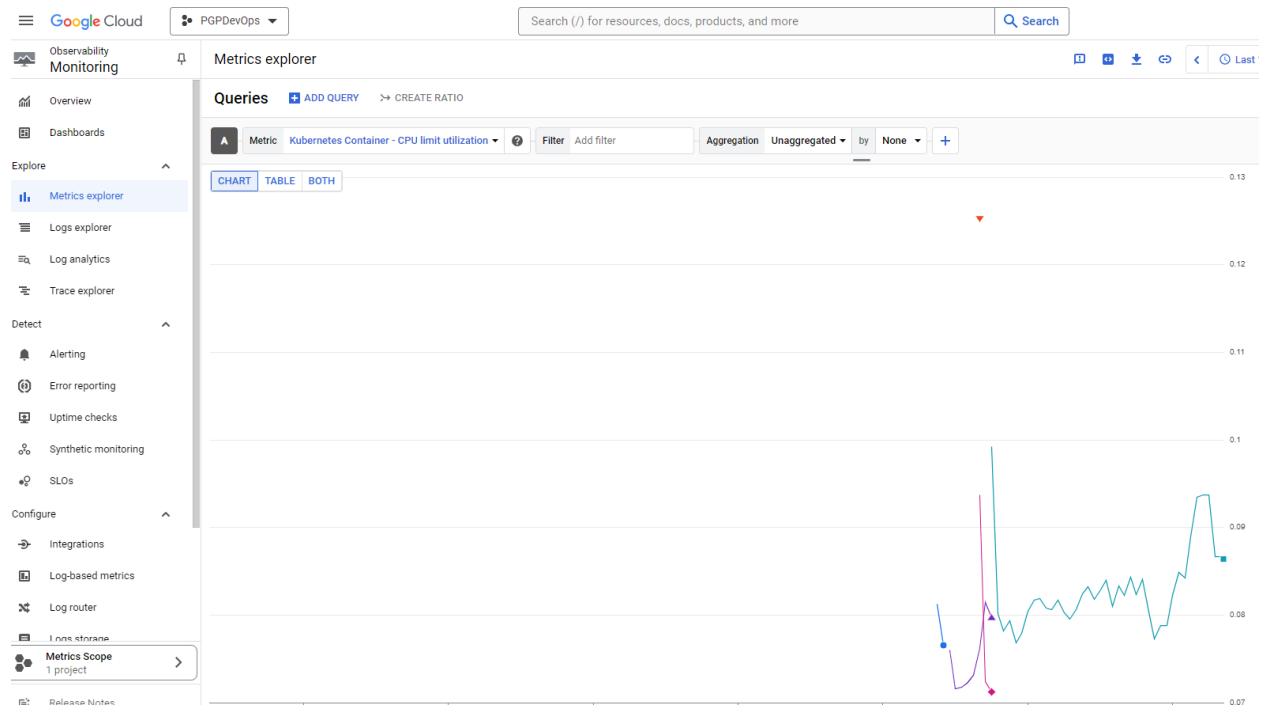
Get up and running with Managed Service for Prometheus

- Enable Managed Service for Prometheus
- Install common exporters and integrations
- Optimize Managed Service for Prometheus metrics for cost control

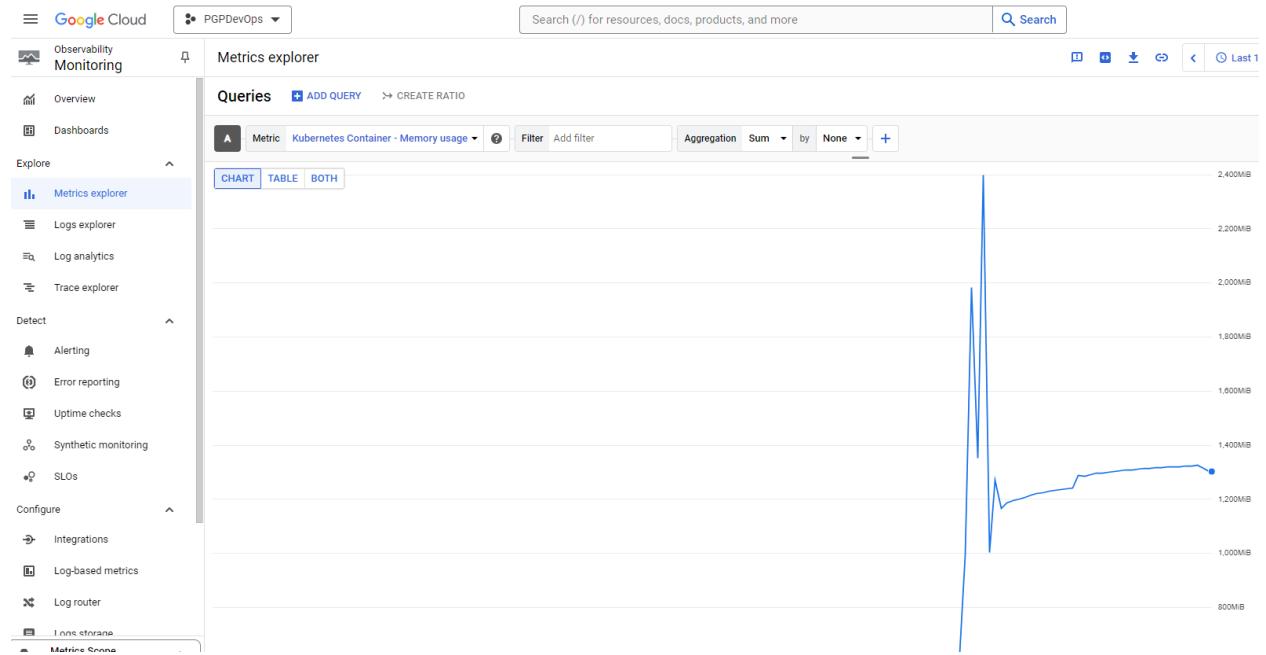
LEARN HOW TO GET STARTED

Categories	Clusters	Managed Prometheus	SELECT ALL AND ENABLE	ENABLE SELECTED	REFRESH	VIEW ALL INTEGRATIONS
Select category	Filter Enter property name or value					
All	Name	Managed Prometheus Status	Ingestion Rate	Location	Mode	
<input checked="" type="checkbox"/> Managed Collection	abctechnologies-pgpr1-cluster	Managed Collection	-	us-east1-b	Standard	
<input type="checkbox"/> Self-Deployed	my-first-cluster-1	Managed Collection	-	us-central1-c	Standard	
<input type="checkbox"/> Not Detected						

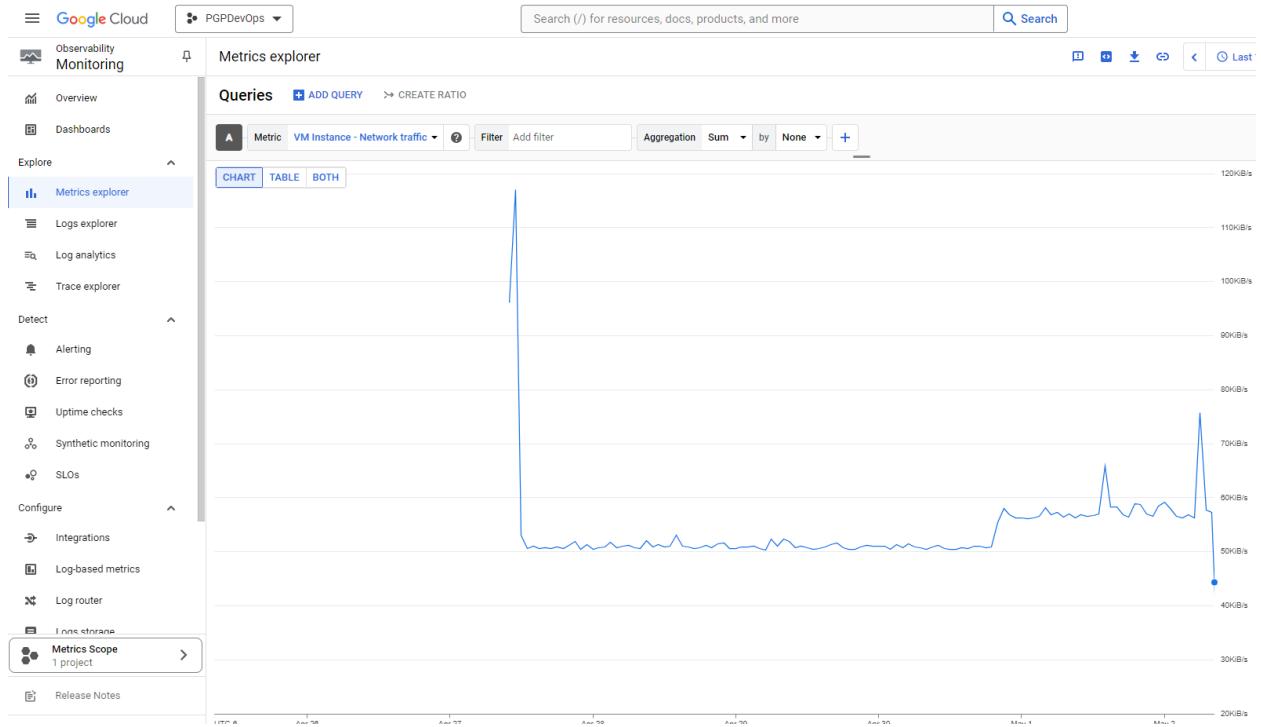
CPU metrics monitoring:



Memory metrics monitoring:

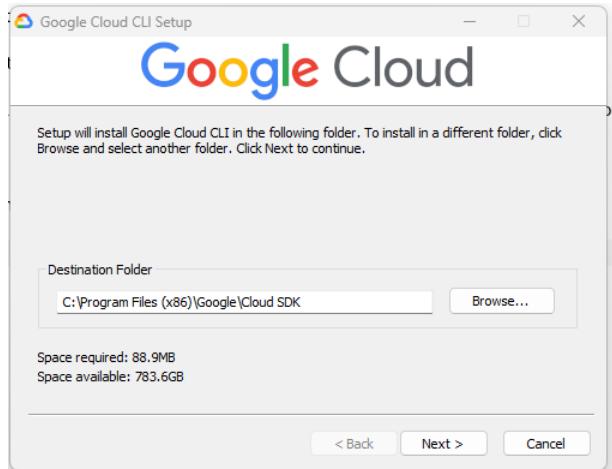


Network traffic from the Kubernetes nodes:



Once I selected resources to monitor, I installed Grafana to translate the logs into easily readable graphs. This would require installing Google Cloud CLI, Node.js and Grafana on my local machine.

Install Google Cloud CLI:



Select which environment you would like to connect to.

```
C:\WINDOWS\SYSTEM32\cmd + - X
Your current configuration has been set to: [default]
You can skip diagnostics next time by using the following flag:
gcloud init --skip-diagnostics

Network diagnostic detects and fixes local network connection issues.
Checking network connection...done.
Reachability Check passed.
Network diagnostic passed (1/1 checks passed).

You must log in to continue. Would you like to log in (Y/n)? Y

Your browser has been opened to visit:

https://accounts.google.com/o/oauth2/auth?response_type=code&client_id=32555940559.apps.googleusercontent.com&redirect_uri=http%3A%2Flocalhost%3A8085%2F&scope=openid+https%3A%2Fwww.googleapis.com%2Fauth%2Fuserinfo.email+https%3A%2Fwww.googleapis.com%2Fauth%2Fcloud-platform+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fappengine.admin+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fsqlservice.login+https%3A%2Fwww.googleapis.com%2Fauth%2Fcompute+https%3A%2Fwww.googleapis.com%2Fauth%2Faccounts.reauth&state=Xih53XNFVAYFA2YY1FSw0u0UUnCZU&access_type=offline&code_challenge=7v-3ozRs7cN9Pf2TLKAb_c8Luirc2jGo9crvcQWzz0&code_challenge_method=S256

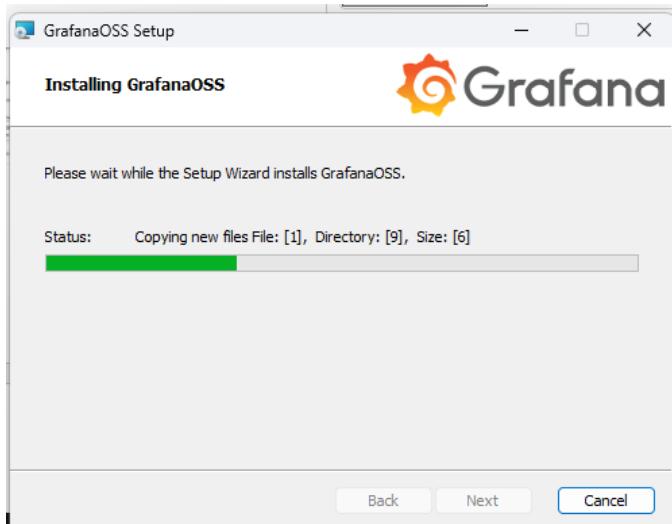
You are logged in as: [charlielong1969@gmail.com].

Pick cloud project to use:
[1] majestic-bounty-420701
[2] pgpdevops
[3] pgpdevops-421717
[4] Enter a project ID
[5] Create a new project
Please enter numeric choice or text value (must exactly match list item):
```

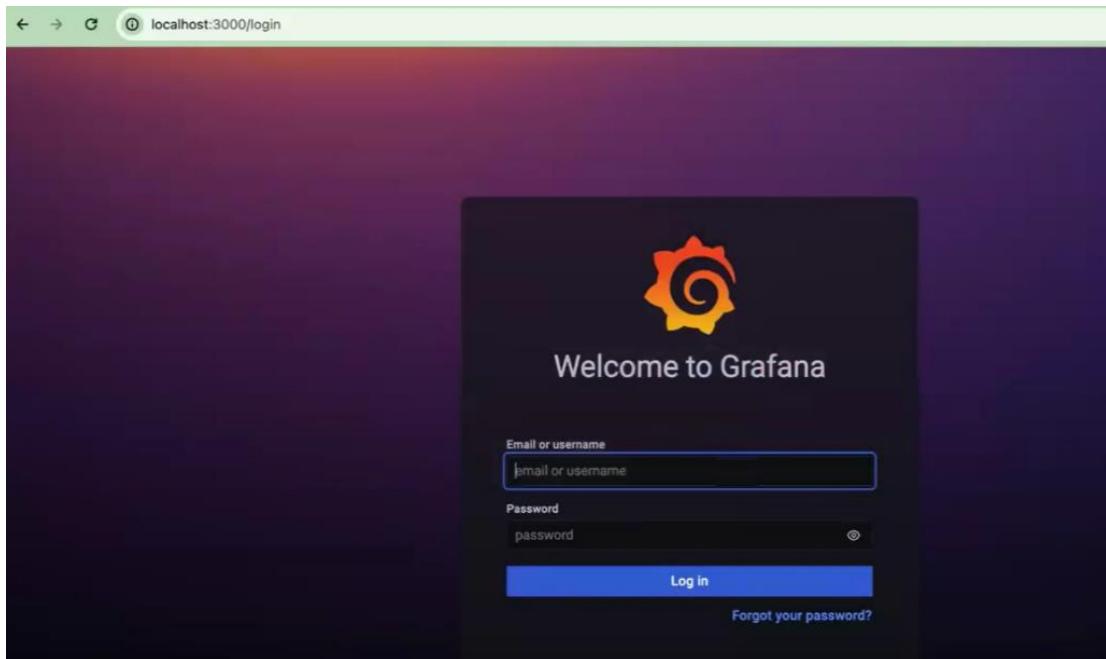
Install Node.js

The screenshot shows the official Node.js website at <https://nodejs.org/en/>. The top navigation bar includes links for Learn, About, Download (which is highlighted in green), Blog, Docs, and Certification. Below the navigation is a large section titled "Download Node.js®". A sub-header says "Download Node.js the way you want.". A horizontal menu bar below the title includes Prebuilt Installer, Prebuilt Binaries, Package Manager, and Source Code. Underneath, there's a dropdown for selecting the Node.js version ("I want the v20.12.2 (LTS) version of Node.js for"), a dropdown for the operating system ("Windows running x64"), and a large green button labeled "Download Node.js v20.12.2". Below the main button are several smaller links: "Node.js includes npm (10.5.0)", "Read the changelog for this version", "Read the blog post for this version", "Learn how to verify signed SHASUMS", "Check out all available Node.js download options", and "Learn about Node.js Releases".

Install Grafana:



Log into Grafana after installation is completed. In order to log into Grafana, users must first forward port in Kubernetes environment to port 3000.



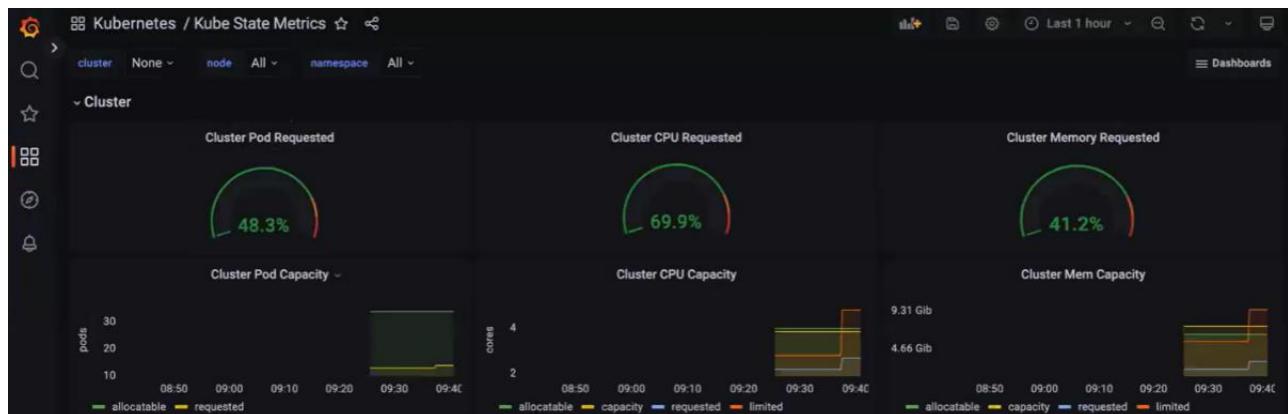
Select your data source as Prometheus.

The screenshot shows the Grafana configuration interface at localhost:3000/datasources. The title bar says "Configuration" and "Organization: Main Org.". The top navigation bar includes links for Data sources, Users, Teams, Plugins, Preferences, API keys, and Service accounts. A search bar is present. Below the navigation, a section titled "Managed Prometheus" is shown, with details: "Prometheus | https://aps-workspaces.us-east-1.amazonaws.com/workspaces/ws-7c3c5930-1137-4200-85ca-be4f440a4f54 | default".

Verify that your data source is working.

This screenshot shows the detailed configuration for the "Managed Prometheus" data source. It includes sections for "Scrape interval" (set to 15s), "Query timeout" (set to 60s), "HTTP Method" (set to POST), "Misc" (with "Disable metrics lookup" turned off), "Custom query parameters" (set to "Example: max_source_reso"), and "Exemplars" (with a "+ Add" button). At the bottom, a green status box indicates "Data source is working" with a checkmark icon.

Finally, view your collected data in a graphical format. Grafana offers many types of displays and widgets, such as gauges, bar charts, line graphs, etc.



Those are all the tasks that were outlined for completion. Overall, this was a challenging project the encompassed the breadth of material that has been covered over the past several months. Most of the issues I had was choosing to use Google as my Kubernetes platform. I was not familiar with the environment and there were things I ran into that were not discussed in class sessions. I probably spent close to 50% of my time on this project figuring out how to configure Google Cloud. Using Amazon probably would have been easier as it was used more heavily during the live courses.

Addendum:

Before deciding to use the GKE service in Google Cloud I deployed two VMs to do a small Kubernetes installation. To do this, I ran the following commands. After completing the process, I ran into multiple issues and opted to go for the provided service from Google.

Building K8s cluster.

Stand up two Ubuntu VMs in Google Cloud.

Connect to each node via SSH.

Commands to create nodes:

Switched to root user by issuing sudo su – command.

Ran following commands on both nodes:

```
apt-get update
```

```
apt-get install -y apt-transport-https ca-certificates curl gpg
```

```
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]  
https://pkgs.k8s.io/core:/stable:/v1.30/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list
```

```
apt-get update  
apt-get install -y kubelet kubeadm kubectl  
apt-mark hold kubelet kubeadm kubectl
```

```
curl -s https://packages.cloud.google.com/apt/doc/yum-key.gpg | apt-key add -  
echo "deb http://apt.kubernetes.io/ kubernetes-xenial main" >/etc/apt/sources.list.d/kubernetes.list  
apt-get update  
apt -y install kubeadm kubectl kubelet  
sed -i 's|disabled_plugin|#disabled_plugin|g' /etc/containerd/config.toml  
service containerd restart
```

Ran following commands on master node only:

```
kubeadm init --ignore-preflight-errors=all
```

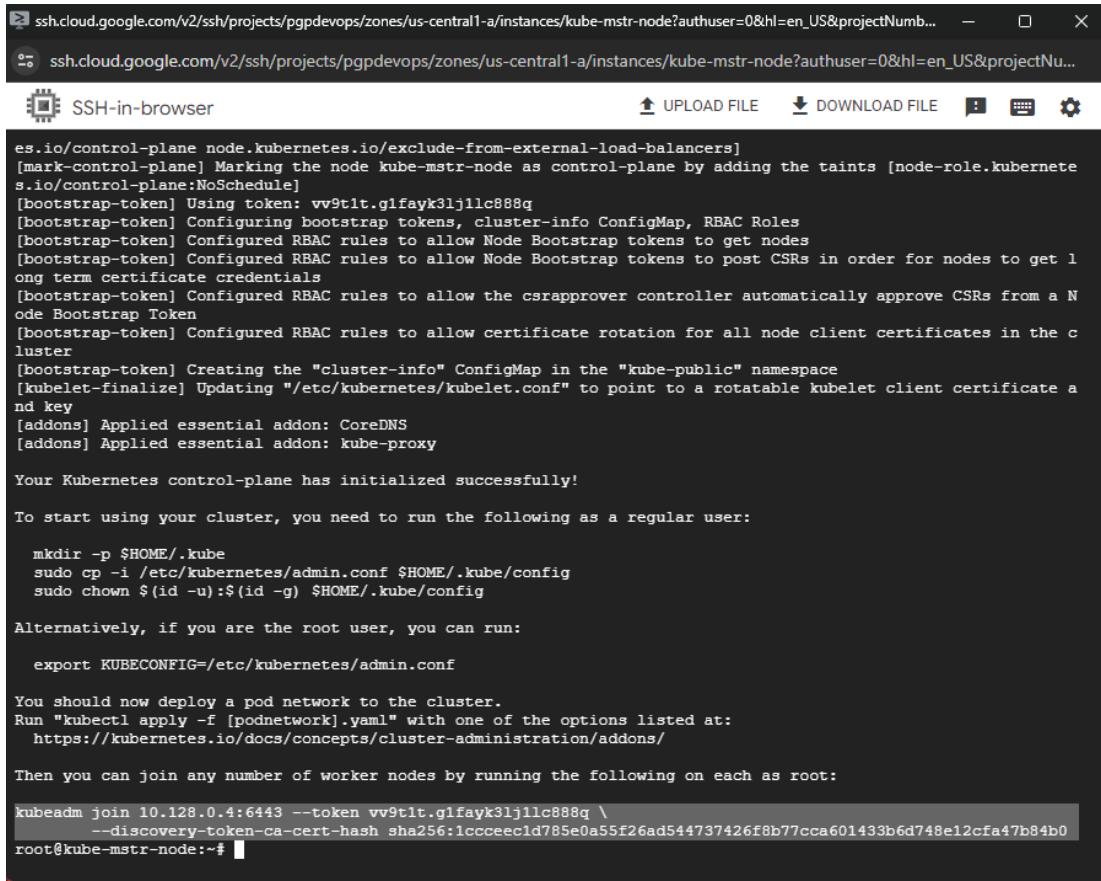
Execute these commands

```
mkdir -p $HOME/.kube  
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config  
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Execute below command to install weave network driver

```
kubectl apply -f https://github.com/weaveworks/weave/releases/download/v2.8.1/weave-daemonset-k8s.yaml
```

After initializing master node, get command to initialize worker node.



The screenshot shows an SSH-in-browser session on a web browser. The title bar indicates the session is connected to a Google Cloud VM instance named 'kube-mstr-node'. The interface includes standard browser controls (back, forward, search) and a toolbar with icons for upload, download, and settings.

The terminal window displays the log output of the node initialization process:

```
[es.io/control-plane node.kubernetes.io/exclude-from-external-load-balancers]
[mark-control-plane] Marking the node kube-mstr-node as control-plane by adding the taints [node-role.kubernetes.io/control-plane:NoSchedule]
[bootstrap-token] Using token: vv9t1t.glfayk3lj1lc888q
[bootstrap-token] Configuring bootstrap tokens, cluster-info ConfigMap, RBAC Roles
[bootstrap-token] Configured RBAC rules to allow Node Bootstrap tokens to get nodes
[bootstrap-token] Configured RBAC rules to allow Node Bootstrap tokens to post CSRs in order for nodes to get long term certificate credentials
[bootstrap-token] Configured RBAC rules to allow the csrapprover controller automatically approve CSRs from a Node Bootstrap Token
[bootstrap-token] Configured RBAC rules to allow certificate rotation for all node client certificates in the cluster
[bootstrap-token] Creating the "cluster-info" ConfigMap in the "kube-public" namespace
[kubelet-finalize] Updating "/etc/kubernetes/kubelet.conf" to point to a rotatable kubelet client certificate and key
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config

Alternatively, if you are the root user, you can run:

export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 10.128.0.4:6443 --token vv9t1t.glfayk3lj1lc888q \
    --discovery-token-ca-cert-hash sha256:1ccceec1d785e0a55f26ad544737426f8b77cca601433b6d748e12cfa47b84b0
root@kube-mstr-node:~#
```

Worker node joined:

```

root@kube-wkr-node:~# service containerd restart
root@kube-wkr-node:~# kubeadm join 10.128.0.4:6443 --token vv9tit.glfayk3ljl1c888q \
>     --discovery-token-ca-cert-hash sha256:1ccceec1d785e0a55f26ad544737426f8b77cca601433b6d748e12cfa47b84b
0
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-check] Waiting for a healthy kubelet. This can take up to 4m0s
[kubelet-check] The kubelet is healthy after 1.002136085s
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.

root@kube-wkr-node:~# 

```

Generate token:

```

Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config

Alternatively, if you are the root user, you can run:

export KUBECONFIG=/etc/kubernetes/admin.conf

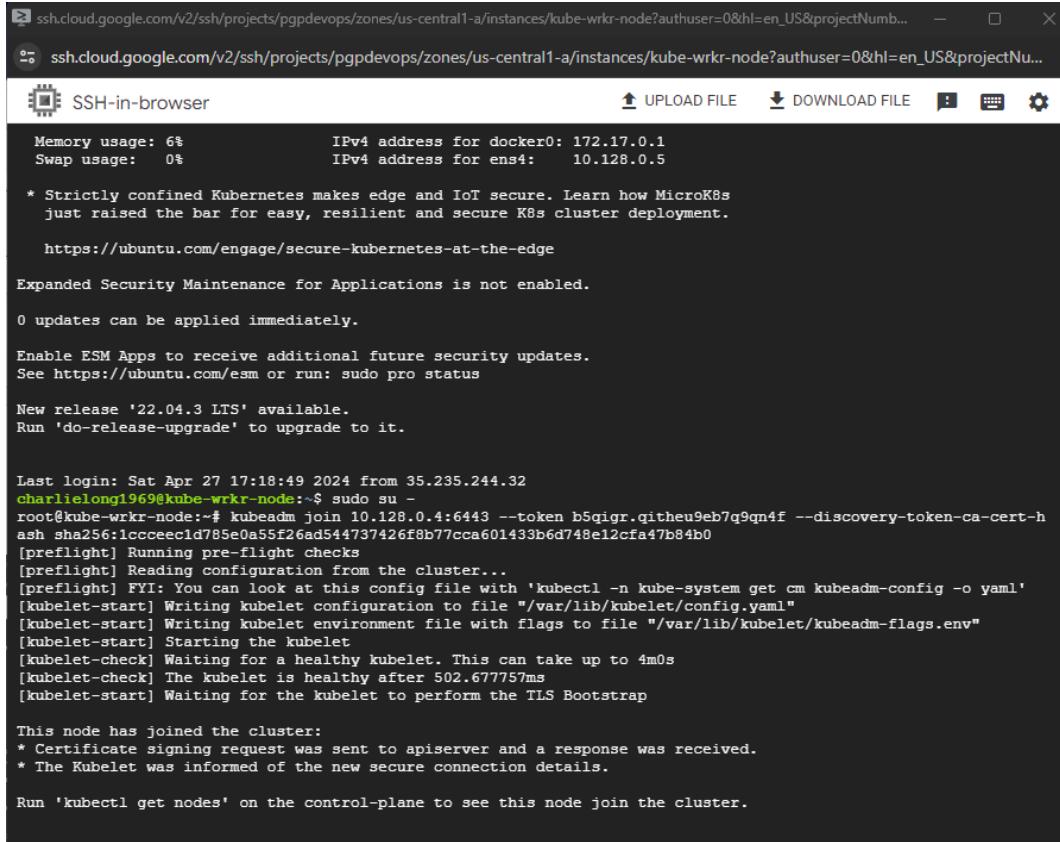
You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 10.128.0.4:6443 --token oea6ou.m4in4lxn04o534eb \
--discovery-token-ca-cert-hash sha256:1ccceec1d785e0a55f26ad544737426f8b77cca601433b6d748e12cfa47b84b0
root@kube-mstr-node:~# mkdir -p $HOME/.kube
root@kube-mstr-node:~# sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
root@kube-mstr-node:~# sudo chown $(id -u):$(id -g) $HOME/.kube/config
root@kube-mstr-node:~# kubectl apply -f https://github.com/weaveworks/weave/releases/download/v2.8.1/weave-daemonset-k8s.yaml
serviceaccount/weave-net created
clusterrole.rbac.authorization.k8s.io/weave-net created
clusterrolebinding.rbac.authorization.k8s.io/weave-net created
role.rbac.authorization.k8s.io/weave-net created
rolebinding.rbac.authorization.k8s.io/weave-net created
daemonset.apps/weave-net created
root@kube-mstr-node:~# kubectl get nodes
NAME           STATUS    ROLES   AGE     VERSION
kube-mstr-node Ready     control-plane   17m    v1.30.0
kube-wkr-node  Ready     <none>    15m    v1.30.0
root@kube-mstr-node:~# kubeadm token create --print-join-command
kubeadm join 10.128.0.4:6443 --token b5qigr.qitheu9eb7q9qn4f --discovery-token-ca-cert-hash sha256:1ccceec1d785e0a55f26ad544737426f8b77cca601433b6d748e12cfa47b84b0
root@kube-mstr-node:~# 

```

After applying token to worker node:



The screenshot shows an SSH-in-browser interface. The terminal window displays a log of a node joining a Kubernetes cluster. The log includes details about memory and swap usage, network interfaces (IPv4 addresses), and a note about strict confinement making edge and IoT secure. It also mentions MicroK8s and provides a link to engage with Kubernetes at the edge. The log continues with expanded security maintenance information, update status, ESM app enablement, and a note about a new LTS release. It concludes with a message about certificate signing and joining the cluster.

```
Memory usage: 6%          IPv4 address for docker0: 172.17.0.1
Swap usage:  0%          IPv4 address for ens4:    10.128.0.5

* Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
just raised the bar for easy, resilient and secure K8s cluster deployment.

https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

New release '22.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Sat Apr 27 17:18:49 2024 from 35.235.244.32
charlielong1969@kube-wrkr-node:~$ sudo su -
root@kube-wrkr-node:~# kubeadm join 10.128.0.4:6443 --token b5qigr.qitheu9eb7q9qn4f --discovery-token-ca-cert-h
ash sha256:1ccceec1d785e0a55f26ad544737426f8b77cce601433b6d748e12cfa47b84b0
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-check] Waiting for a healthy kubelet. This can take up to 4m0s
[kubelet-check] The kubelet is healthy after 502.677757ms
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
```

```
charlielong1969@cloudshell:~ (pgpdevops)$ kubectl get nodes
NAME                  STATUS   ROLES      AGE     VERSION
gke-my-first-cluster-1-default-pool-fd1ec955-g58w  Ready    <none>   2d1h   v1.29.3-gke.1093000
gke-my-first-cluster-1-default-pool-fd1ec955-sh4v  Ready    <none>   2d1h   v1.29.3-gke.1093000
gke-my-first-cluster-1-default-pool-fd1ec955-wbcn  Ready    <none>   14h    v1.29.3-gke.1093000
```

Pods running in Kubernetes: Issued kubectl command to show running pods.

```
charlielong1969@cloudshell:~ (pgpdevops)$ kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
abctech-pgp-67b8b99f79-f6czn  1/1     Running   0          27m
abctech-pgp-67b8b99f79-jqb9d  1/1     Running   0          27m
abctech-pgp-67b8b99f79-vdjk7  1/1     Running   0          27m
charlielong1969@cloudshell:~ (pgpdevops)$
```

For Jenkins to work with the various cloud services, it required several credentials to be configured:

The screenshot shows the Jenkins 'Credentials' page. At the top, there's a navigation bar with 'Dashboard > Manage Jenkins > Credentials'. Below the header, the title 'Credentials' is centered. A table lists five global credentials:

T	P	Store	Domain	ID	Name
System	System	(global)	dockerhub	clong1969/******** (dockerhub)	
System	System	(global)	gke	charlieclong1969@gmail.com/******** (google cloud)	
System	System	(global)	github	clong1969/******** (github)	
System	System	(global)	pgpjenkins	PGPDevOps	
System	System	(global)	pgpjenkins2	pgpdevops	

Credentials

T	P	Store	Domain	ID	Name
System	System	(global)	dockerhub	clong1969/******** (dockerhub)	
System	System	(global)	gke	charlieclong1969@gmail.com/******** (google cloud)	
System	System	(global)	github	clong1969/******** (github)	
System	System	(global)	pgpjenkins	PGPDevOps	
System	System	(global)	pgpjenkins2	pgpdevops	

Stores scoped to Jenkins

P	Store	Domains
System	System	(global)
Kubernetes	Kubernetes	(global)

Example with GitHub credentials.

The screenshot shows the 'Update credentials' form for a GitHub credential. The URL in the browser is 'Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) > clong1969/******** (github)'. The form fields are as follows:

- Update**: Action button.
- Delete**: Delete button.
- Move**: Move button.
- Scope**: Set to 'Global (Jenkins, nodes, items, all child items, etc)'.
- Username**: Value 'clong1969'.
- Treat username as secret**: Unchecked checkbox.
- Password**: Value 'Concealed' (password field).
- ID**: Value 'github'.
- Description**: Value 'github'.
- Save**: Primary action button.

In order to interact with Google Cloud, and cloud provider needed to be configured as well.

 Jenkins

Dashboard > Manage Jenkins > Clouds >

Clouds

During node provisioning, clouds are tried in the order they appear in this table.

Order	Name
1	 pgpdevops

 Jenkins

Dashboard > Manage Jenkins > Clouds > pgpdevops > Configure

Status Pod Templates Configure Delete Cloud

Cloud pgpdevops Configuration

Name

Kubernetes Cloud details

Kubernetes URL

Use Jenkins Proxy

Kubernetes server certificate key

Disable https certificate check