AI Planet

DevOps Internship Assignment

Aim

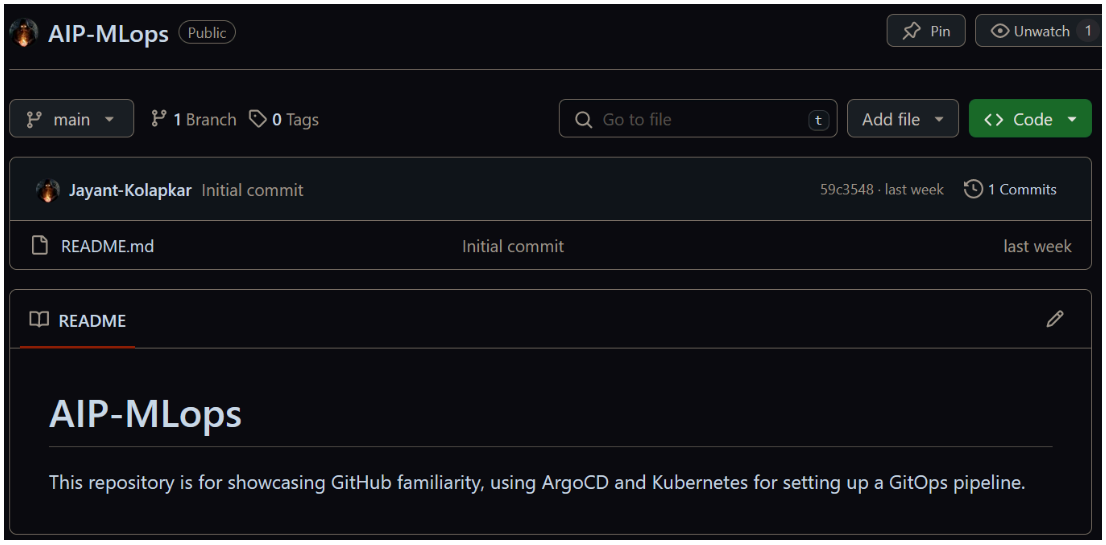
The aim of this assignment is to understand the hands-on experience with GitOps practices, utilizing Argo CD for continuous deployment and Argo Rollouts for advanced deployment strategies within a Kubernetes environment.
Here I am using WSL with Ubuntu 22.04 as my environment. I encountered a barrier which I was unable to cross, and tried to replicate the process in Windows with no avail. So I just documented the remaining steps with instructions without any screenshots

Steps

Task 1: Setup and Configuration

Step 1: Create a GitRepository

1. Go to GitHub and create a new repository to host your source code.
2. You can either initialize the repository with a README file or create it empty.
3. This repository will serve as the single source of truth for your application's code, Kubernetes manifests, and other configuration files.



Jayant-Kolapkar Initial commit 59c3548 · last week 1 Commits

README.md Initial commit last week

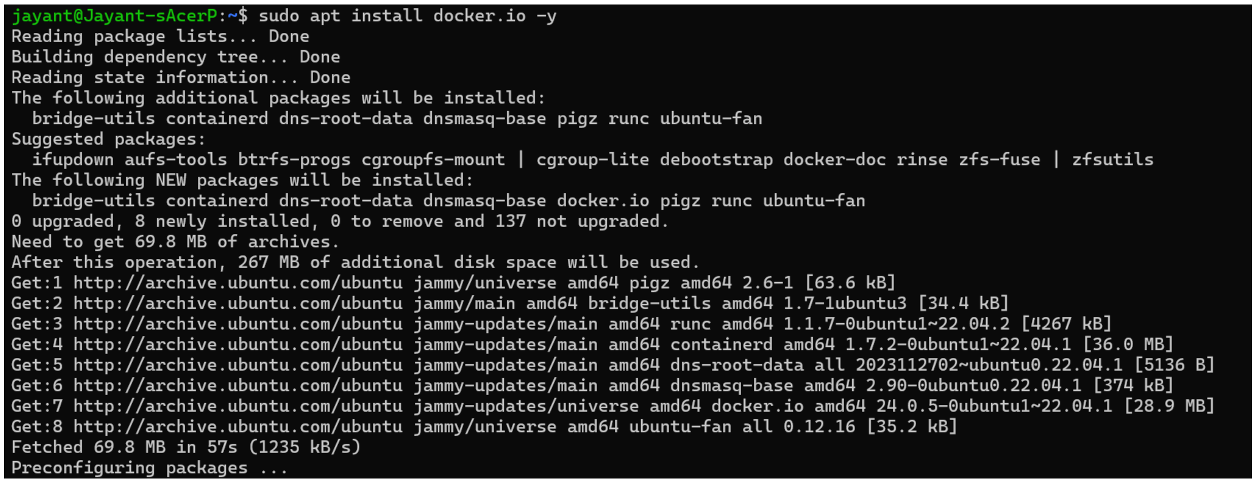
AIP-MLops

This repository is for showcasing GitHub familiarity, using ArgoCD and Kubernetes for setting up a GitOps pipeline.

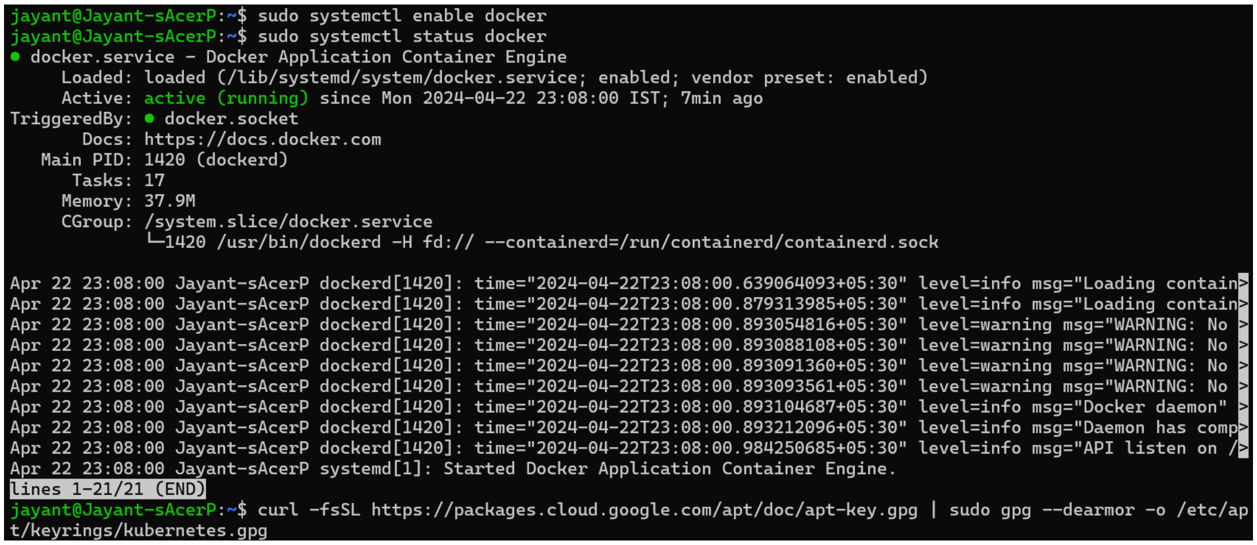
Step 2: Install Docker, kubeadm and kubectl

All of these pre-requisites are needed to run ArgoCD.

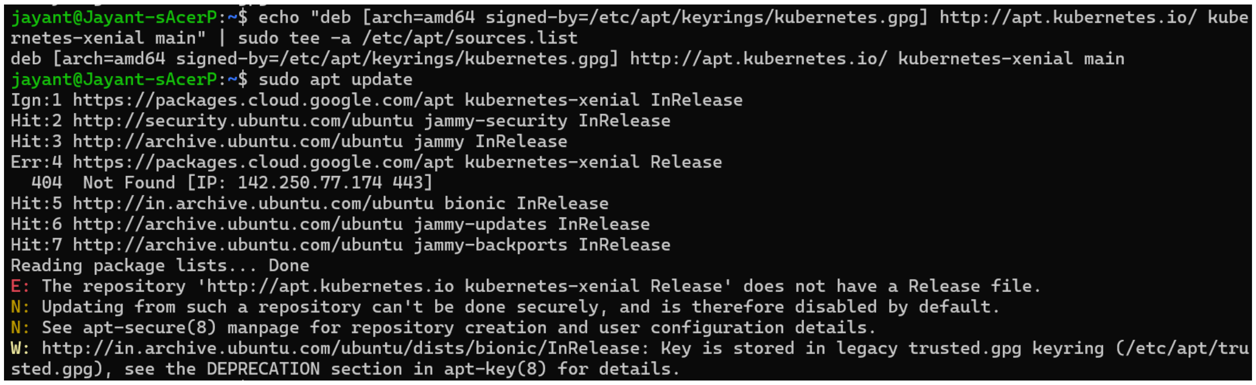
Installing Docker



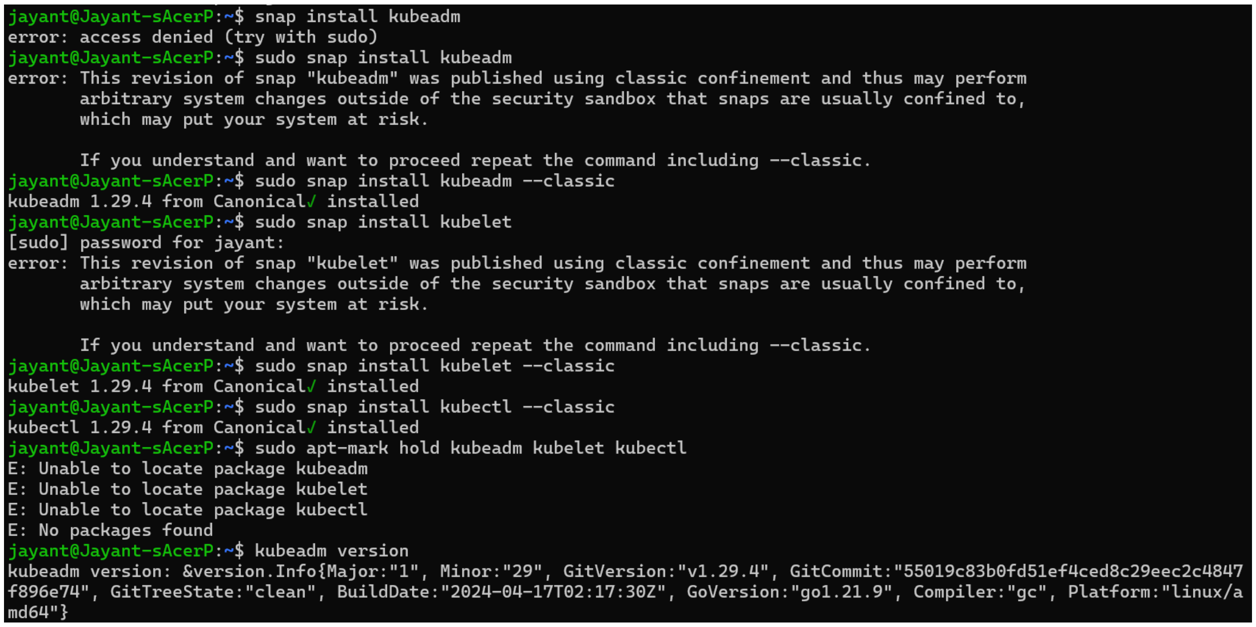
```
jayant@Jayant-sAcerP:~$ sudo apt install docker.io -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools btrfs-progs 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 pigz runc ubuntu-fan
0 upgraded, 8 newly installed, 0 to remove and 137 not upgraded.
Need to get 69.8 MB of archives.
After this operation, 267 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy/universe amd64 pigz amd64 2.6-1 [63.6 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy/main amd64 bridge-utils amd64 1.7-1ubuntu3 [34.4 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 runc amd64 1.1.7-0ubuntu1~22.04.2 [4267 kB]
Get:4 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 containerd amd64 1.7.2-0ubuntu1~22.04.1 [36.0 MB]
Get:5 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 dns-root-data all 2023112702~ubuntu0.22.04.1 [5136 B]
Get:6 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 dnsmasq-base amd64 2.90-0ubuntu0.22.04.1 [374 kB]
Get:7 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 docker.io amd64 24.0.5-0ubuntu1~22.04.1 [28.9 MB]
Get:8 http://archive.ubuntu.com/ubuntu jammy/universe amd64 ubuntu-fan all 0.12.16 [35.2 kB]
Fetched 69.8 MB in 57s (1235 kB/s)
Preconfiguring packages ...
```



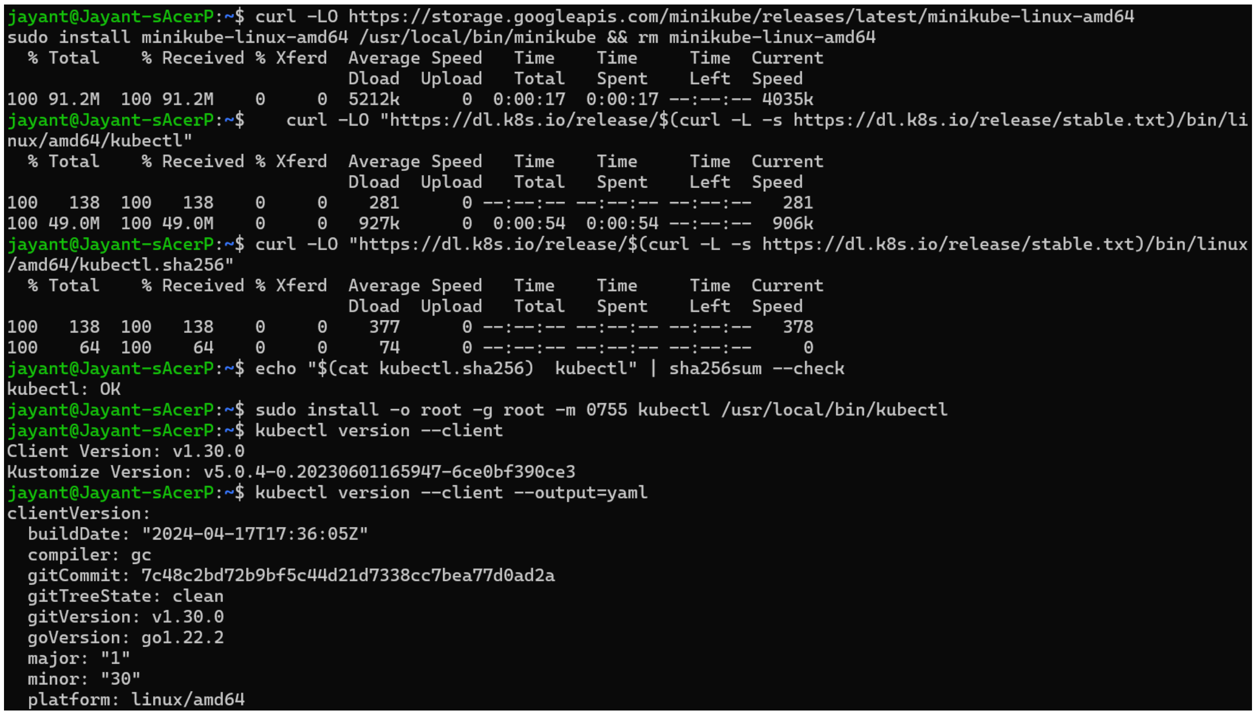
```
jayant@Jayant-sAcerP:~$ sudo systemctl enable docker
jayant@Jayant-sAcerP:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
     Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
     Active: active (running) since Mon 2024-04-22 23:08:00 IST; 7min ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 1420 (dockerd)
      Tasks: 17
     Memory: 37.9M
     CGroup: /system.slice/docker.service
             └─1420 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Apr 22 23:08:00 Jayant-sAcerP dockerd[1420]: time="2024-04-22T23:08:00.639064093+05:30" level=info msg="Loading contain>
Apr 22 23:08:00 Jayant-sAcerP dockerd[1420]: time="2024-04-22T23:08:00.879313985+05:30" level=info msg="Loading contain>
Apr 22 23:08:00 Jayant-sAcerP dockerd[1420]: time="2024-04-22T23:08:00.893054816+05:30" level=warning msg="WARNING: No >
Apr 22 23:08:00 Jayant-sAcerP dockerd[1420]: time="2024-04-22T23:08:00.893088108+05:30" level=warning msg="WARNING: No >
Apr 22 23:08:00 Jayant-sAcerP dockerd[1420]: time="2024-04-22T23:08:00.893091360+05:30" level=warning msg="WARNING: No >
Apr 22 23:08:00 Jayant-sAcerP dockerd[1420]: time="2024-04-22T23:08:00.893093561+05:30" level=warning msg="WARNING: No >
Apr 22 23:08:00 Jayant-sAcerP dockerd[1420]: time="2024-04-22T23:08:00.893104687+05:30" level=info msg="Docker daemon" >
Apr 22 23:08:00 Jayant-sAcerP dockerd[1420]: time="2024-04-22T23:08:00.893212096+05:30" level=info msg="Daemon has comp>
Apr 22 23:08:00 Jayant-sAcerP dockerd[1420]: time="2024-04-22T23:08:00.984250685+05:30" level=info msg="API listen on />
Apr 22 23:08:00 Jayant-sAcerP systemd[1]: Started Docker Application Container Engine.
lines 1-21/21 (END)
jayant@Jayant-sAcerP:~$ curl -fsSL https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo gpg --dearmor -o /etc/ap
t/keyrings/kubernetes.gpg
```


```
jayant@Jayant-sAcerP:~$ echo "deb [arch=amd64 signed-by=/etc/apt/keyrings/kubernetes.gpg] http://apt.kubernetes.io/ kube
rnetes-xenial main" | sudo tee -a /etc/apt/sources.list
deb [arch=amd64 signed-by=/etc/apt/keyrings/kubernetes.gpg] http://apt.kubernetes.io/ kubernetes-xenial main
jayant@Jayant-sAcerP:~$ sudo apt update
Ign:1 https://packages.cloud.google.com/apt kubernetes-xenial InRelease
Hit:2 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:3 http://archive.ubuntu.com/ubuntu jammy InRelease
Err:4 https://packages.cloud.google.com/apt kubernetes-xenial Release
  404  Not Found [IP: 142.250.77.174 443]
Hit:5 http://in.archive.ubuntu.com/ubuntu bionic InRelease
Hit:6 http://archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:7 http://archive.ubuntu.com/ubuntu jammy-backports InRelease
Reading package lists... Done
E: The repository 'http://apt.kubernetes.io kubernetes-xenial Release' does not have a Release file.
N: Updating from such a repository can't be done securely, and is therefore disabled by default.
N: See apt-secure(8) manpage for repository creation and user configuration details.
W: http://in.archive.ubuntu.com/ubuntu/dists/bionic/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt/tru
sted.gpg), see the DEPRECATION section in apt-key(8) for details.
```

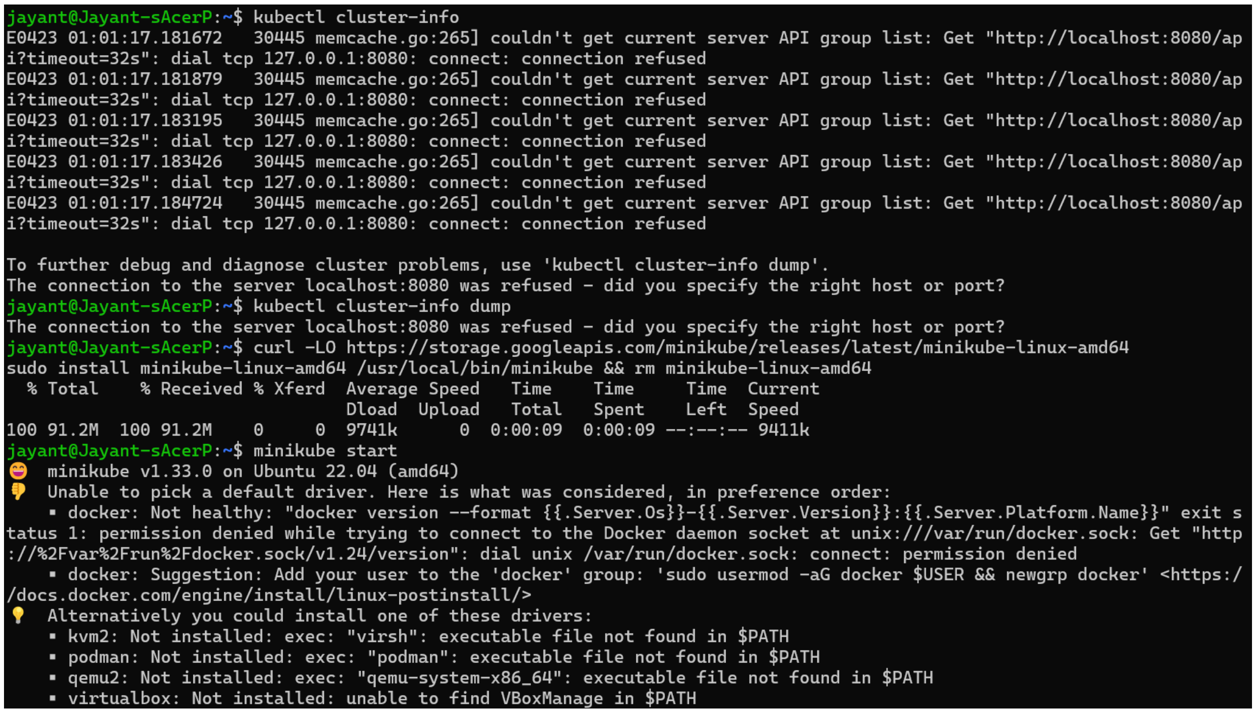
Installing kubeadm


```
jayant@Jayant-sAcerP:~$ snap install kubeadm
error: access denied (try with sudo)
jayant@Jayant-sAcerP:~$ sudo snap install kubeadm
error: This revision of snap "kubeadm" was published using classic confinement and thus may perform
       arbitrary system changes outside of the security sandbox that snaps are usually confined to,
       which may put your system at risk.

       If you understand and want to proceed repeat the command including --classic.
jayant@Jayant-sAcerP:~$ sudo snap install kubeadm --classic
kubeadm 1.29.4 from Canonical✓ installed
jayant@Jayant-sAcerP:~$ sudo snap install kubelet
[sudo] password for jayant:
error: This revision of snap "kubelet" was published using classic confinement and thus may perform
       arbitrary system changes outside of the security sandbox that snaps are usually confined to,
       which may put your system at risk.

       If you understand and want to proceed repeat the command including --classic.
jayant@Jayant-sAcerP:~$ sudo snap install kubelet --classic
kubelet 1.29.4 from Canonical✓ installed
jayant@Jayant-sAcerP:~$ sudo snap install kubectl --classic
kubectl 1.29.4 from Canonical✓ installed
jayant@Jayant-sAcerP:~$ sudo apt-mark hold kubeadm kubelet kubectl
E: Unable to locate package kubeadm
E: Unable to locate package kubelet
E: Unable to locate package kubectl
E: No packages found
jayant@Jayant-sAcerP:~$ kubeadm version
kubeadm version: &version.Info{Major:"1", Minor:"29", GitVersion:"v1.29.4", GitCommit:"55019c83b0fd51ef4ced8c29eec2c4847
f896e74", GitTreeState:"clean", BuildDate:"2024-04-17T02:17:30Z", GoVersion:"go1.21.9", Compiler:"gc", Platform:"linux/a
md64"}
```

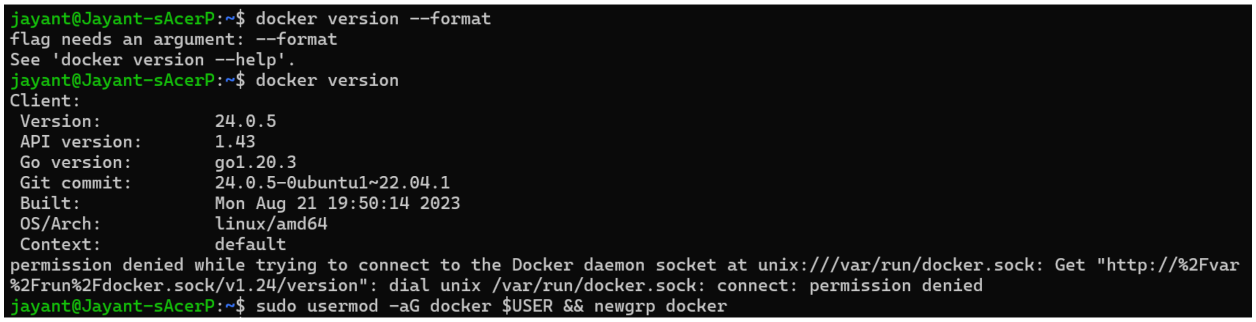
Installing kubectl



```
jayant@Jayant-sAcerP:~$ curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
sudo install minikube-linux-amd64 /usr/local/bin/minikube && rm minikube-linux-amd64
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 91.2M  100 91.2M    0     0  5212k      0  0:00:17  0:00:17 --:--:-- 4035k
jayant@Jayant-sAcerP:~$    curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/li
nux/amd64/kubectl"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   138  100   138    0     0    281      0 --:--:-- --:--:-- --:--:--   281
100 49.0M  100 49.0M    0     0   927k      0  0:00:54  0:00:54 --:--:--  906k
jayant@Jayant-sAcerP:~$ curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux
/amd64/kubectl.sha256"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   138  100   138    0     0    377      0 --:--:-- --:--:-- --:--:--   378
100    64  100    64    0     0     74      0 --:--:-- --:--:-- --:--:--     0
jayant@Jayant-sAcerP:~$ echo "$(cat kubectl.sha256)  kubectl" | sha256sum --check
kubectl: OK
jayant@Jayant-sAcerP:~$ sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
jayant@Jayant-sAcerP:~$ kubectl version --client
Client Version: v1.30.0
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
jayant@Jayant-sAcerP:~$ kubectl version --client --output=yaml
clientVersion:
  buildDate: "2024-04-17T17:36:05Z"
  compiler: gc
  gitCommit: 7c48c2bd72b9bf5c44d21d7338cc7bea77d0ad2a
  gitTreeState: clean
  gitVersion: v1.30.0
  goVersion: go1.22.2
  major: "1"
  minor: "30"
  platform: linux/amd64
```

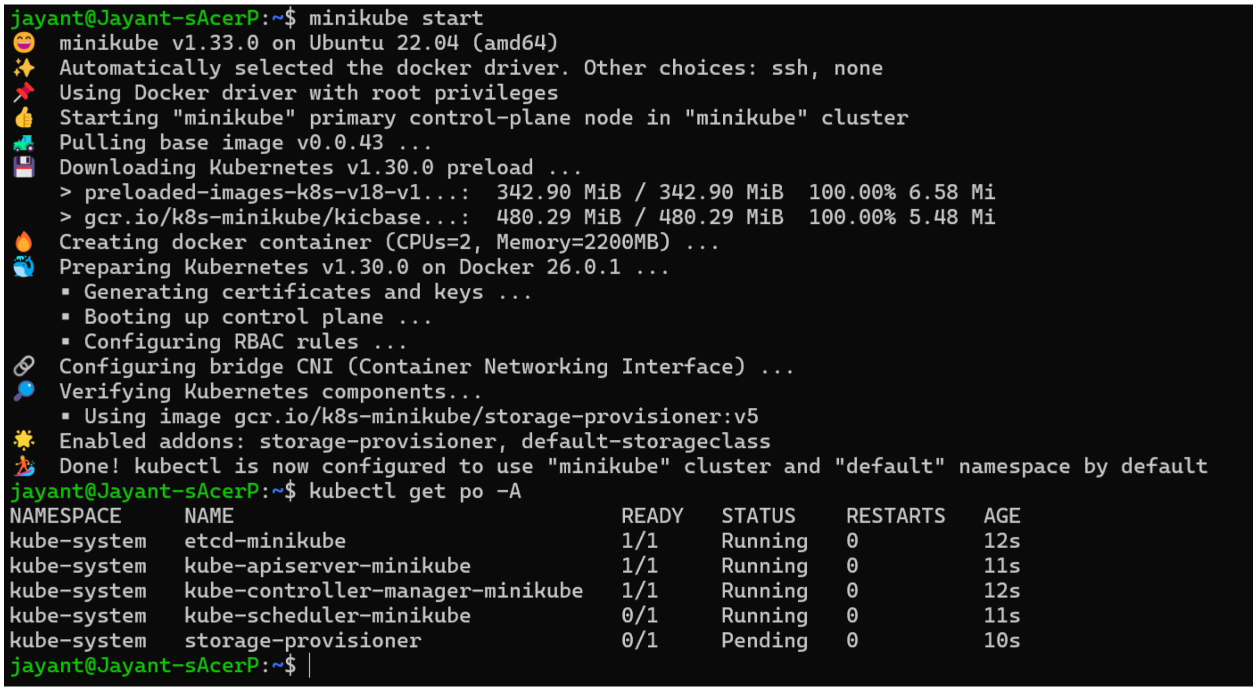


```
jayant@Jayant-sAcerP:~$ kubectl cluster-info
E0423 01:01:17.181672   30445 memcache.go:265] couldn't get current server API group list: Get "http://localhost:8080/ap
i?timeout=32s": dial tcp 127.0.0.1:8080: connect: connection refused
E0423 01:01:17.181879   30445 memcache.go:265] couldn't get current server API group list: Get "http://localhost:8080/ap
i?timeout=32s": dial tcp 127.0.0.1:8080: connect: connection refused
E0423 01:01:17.183195   30445 memcache.go:265] couldn't get current server API group list: Get "http://localhost:8080/ap
i?timeout=32s": dial tcp 127.0.0.1:8080: connect: connection refused
E0423 01:01:17.183426   30445 memcache.go:265] couldn't get current server API group list: Get "http://localhost:8080/ap
i?timeout=32s": dial tcp 127.0.0.1:8080: connect: connection refused
E0423 01:01:17.184724   30445 memcache.go:265] couldn't get current server API group list: Get "http://localhost:8080/ap
i?timeout=32s": dial tcp 127.0.0.1:8080: connect: connection refused

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
The connection to the server localhost:8080 was refused - did you specify the right host or port?
jayant@Jayant-sAcerP:~$ kubectl cluster-info dump
The connection to the server localhost:8080 was refused - did you specify the right host or port?
jayant@Jayant-sAcerP:~$ curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
sudo install minikube-linux-amd64 /usr/local/bin/minikube && rm minikube-linux-amd64
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 91.2M  100 91.2M    0     0  9741k      0  0:00:09  0:00:09 --:--:-- 9411k
jayant@Jayant-sAcerP:~$ minikube start
😄  minikube v1.33.0 on Ubuntu 22.04 (amd64)
👎  Unable to pick a default driver. Here is what was considered, in preference order:
    ▪ docker: Not healthy: "docker version --format {{.Server.Os}}-{{.Server.Version}}:{{.Server.Platform.Name}}" exit s
tatus 1: permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get "http
://%2Fvar%2Frun%2Fdocker.sock/v1.24/version": dial unix /var/run/docker.sock: connect: permission denied
    ▪ docker: Suggestion: Add your user to the 'docker' group: 'sudo usermod -aG docker $USER && newgrp docker' <https:/
/docs.docker.com/engine/install/linux-postinstall/>
💡  Alternatively you could install one of these drivers:
    ▪ kvm2: Not installed: exec: "virsh": executable file not found in $PATH
    ▪ podman: Not installed: exec: "podman": executable file not found in $PATH
    ▪ qemu2: Not installed: exec: "qemu-system-x86_64": executable file not found in $PATH
    ▪ virtualbox: Not installed: unable to find VBoxManage in $PATH
```

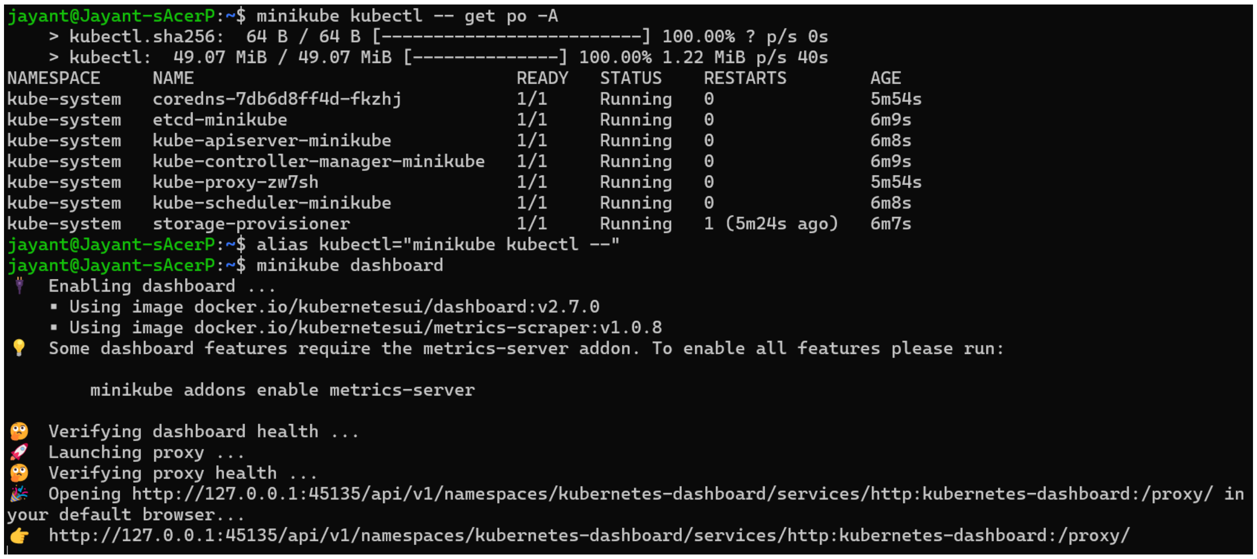


```
jayant@Jayant-sAcerP:~$ docker version --format
flag needs an argument: --format
See 'docker version --help'.
jayant@Jayant-sAcerP:~$ docker version
Client:
 Version:           24.0.5
 API version:       1.43
 Go version:        go1.20.3
 Git commit:        24.0.5-0ubuntu1~22.04.1
 Built:             Mon Aug 21 19:50:14 2023
 OS/Arch:           linux/amd64
 Context:           default
permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get "http://%2Fvar
%2Frun%2Fdocker.sock/v1.24/version": dial unix /var/run/docker.sock: connect: permission denied
jayant@Jayant-sAcerP:~$ sudo usermod -aG docker $USER && newgrp docker
```

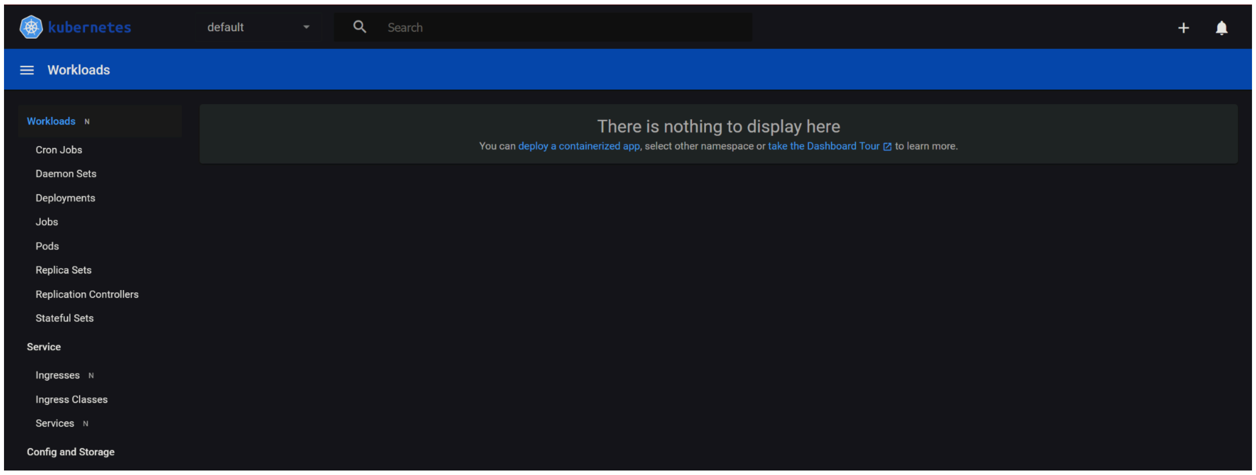
Running minikube



```
jayant@Jayant-sAcerP:~$ minikube start
😄  minikube v1.33.0 on Ubuntu 22.04 (amd64)
✨  Automatically selected the docker driver. Other choices: ssh, none
📌  Using Docker driver with root privileges
👍  Starting "minikube" primary control-plane node in "minikube" cluster
🚜  Pulling base image v0.0.43 ...
💾  Downloading Kubernetes v1.30.0 preload ...
    > preloaded-images-k8s-v18-v1...:  342.90 MiB / 342.90 MiB  100.00% 6.58 Mi
    > gcr.io/k8s-minikube/kicbase...:  480.29 MiB / 480.29 MiB  100.00% 5.48 Mi
🔥  Creating docker container (CPUs=2, Memory=2200MB) ...
🐳  Preparing Kubernetes v1.30.0 on Docker 26.0.1 ...
    ▪ Generating certificates and keys ...
    ▪ Booting up control plane ...
    ▪ Configuring RBAC rules ...
🔗  Configuring bridge CNI (Container Networking Interface) ...
🔎  Verifying Kubernetes components...
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
🌟  Enabled addons: storage-provisioner, default-storageclass
🏄  Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
jayant@Jayant-sAcerP:~$ kubectl get po -A
NAMESPACE     NAME                               READY   STATUS    RESTARTS   AGE
kube-system   etcd-minikube                      1/1     Running   0          12s
kube-system   kube-apiserver-minikube            1/1     Running   0          11s
kube-system   kube-controller-manager-minikube   1/1     Running   0          12s
kube-system   kube-scheduler-minikube            0/1     Running   0          11s
kube-system   storage-provisioner                0/1     Pending   0          10s
jayant@Jayant-sAcerP:~$ 
```



```
jayant@Jayant-sAcerP:~$ minikube kubectl -- get po -A
    > kubectl.sha256:  64 B / 64 B [-------------------------] 100.00% ? p/s 0s
    > kubectl:  49.07 MiB / 49.07 MiB [--------------] 100.00% 1.22 MiB p/s 40s
NAMESPACE     NAME                               READY   STATUS    RESTARTS        AGE
kube-system   coredns-7db6d8ff4d-fkzhj           1/1     Running   0               5m54s
kube-system   etcd-minikube                      1/1     Running   0               6m9s
kube-system   kube-apiserver-minikube            1/1     Running   0               6m8s
kube-system   kube-controller-manager-minikube   1/1     Running   0               6m9s
kube-system   kube-proxy-zw7sh                   1/1     Running   0               5m54s
kube-system   kube-scheduler-minikube            1/1     Running   0               6m8s
kube-system   storage-provisioner                1/1     Running   1 (5m24s ago)   6m7s
jayant@Jayant-sAcerP:~$ alias kubectl="minikube kubectl --"
jayant@Jayant-sAcerP:~$ minikube dashboard
🔌  Enabling dashboard ...
    ▪ Using image docker.io/kubernetesui/dashboard:v2.7.0
    ▪ Using image docker.io/kubernetesui/metrics-scraper:v1.0.8
💡  Some dashboard features require the metrics-server addon. To enable all features please run:

        minikube addons enable metrics-server

🤔  Verifying dashboard health ...
🚀  Launching proxy ...
🤔  Verifying proxy health ...
🎉  Opening http://127.0.0.1:45135/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/ in
your default browser...
👉  http://127.0.0.1:45135/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/
```



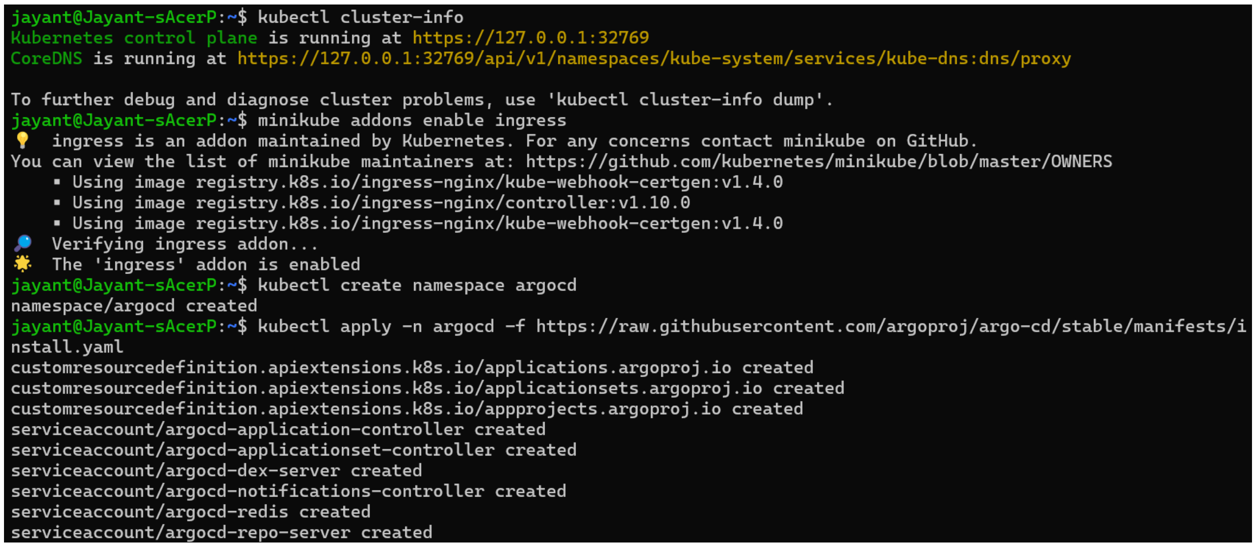
There is nothing to display here

You can deploy a containerized app, select other namespace or take the Dashboard Tour to learn more.

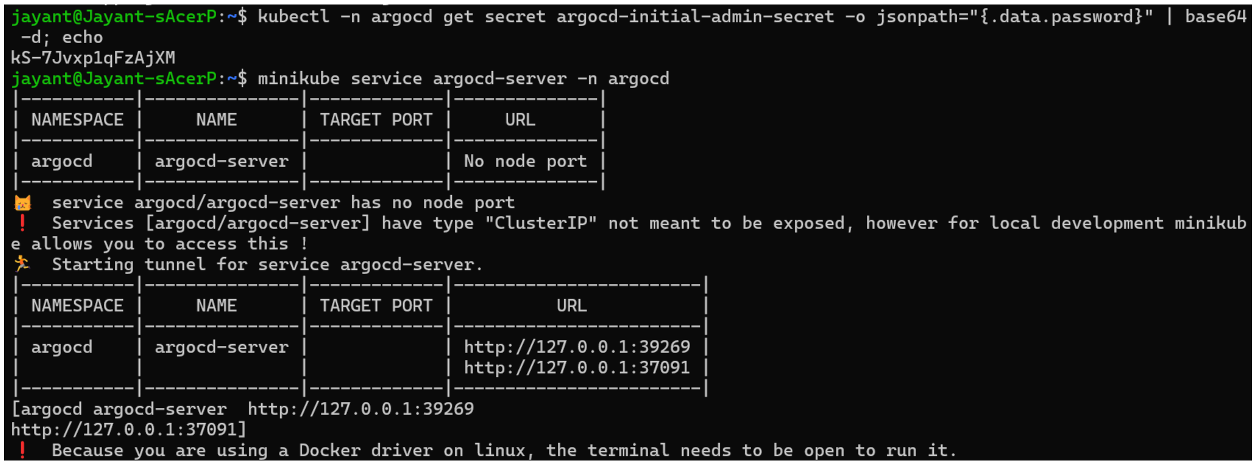
Step 3: Install Argo CD on Your Kubernetes Cluster

1. Argo CD is a declarative continuous deployment tool for Kubernetes.
2. Follow the official [Argo CD installation guide]() to install it on your Kubernetes cluster.
3. The installation process involves creating a namespace, installing the Argo CD components (controller, server, repo-server, etc.), and exposing the Argo CD API server.
4. After installation, you'll have access to the Argo CD UI and CLI for managing your GitOps workflow.

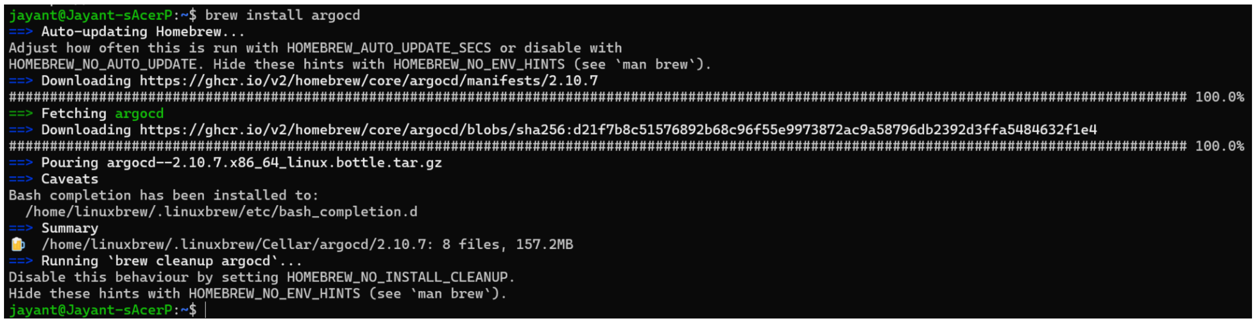
Installing ArgoCD



```
jayant@Jayant-sAcerP:~$ kubectl cluster-info
Kubernetes control plane is running at https://127.0.0.1:32769
CoreDNS is running at https://127.0.0.1:32769/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
jayant@Jayant-sAcerP:~$ minikube addons enable ingress
💡  ingress is an addon maintained by Kubernetes. For any concerns contact minikube on GitHub.
You can view the list of minikube maintainers at: https://github.com/kubernetes/minikube/blob/master/OWNERS
    ▪ Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v1.4.0
    ▪ Using image registry.k8s.io/ingress-nginx/controller:v1.10.0
    ▪ Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v1.4.0
🔎  Verifying ingress addon...
🌟  The 'ingress' addon is enabled
jayant@Jayant-sAcerP:~$ kubectl create namespace argocd
namespace/argocd created
jayant@Jayant-sAcerP:~$ kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/i
nstall.yaml
customresourcedefinition.apiextensions.k8s.io/applications.argoproj.io created
customresourcedefinition.apiextensions.k8s.io/applicationsets.argoproj.io created
customresourcedefinition.apiextensions.k8s.io/appprojects.argoproj.io created
serviceaccount/argocd-application-controller created
serviceaccount/argocd-applicationset-controller created
serviceaccount/argocd-dex-server created
serviceaccount/argocd-notifications-controller created
serviceaccount/argocd-redis created
serviceaccount/argocd-repo-server created
```



```
jayant@Jayant-sAcerP:~$ kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath="{.data.password}" | base64
 -d; echo
kS-7Jvxp1qFzAjXM
jayant@Jayant-sAcerP:~$ minikube service argocd-server -n argocd
|-----------|---------------|-------------|--------------|
| NAMESPACE |     NAME      | TARGET PORT |     URL      |
|-----------|---------------|-------------|--------------|
| argocd    | argocd-server |             | No node port |
|-----------|---------------|-------------|--------------|
😿  service argocd/argocd-server has no node port
❗  Services [argocd/argocd-server] have type "ClusterIP" not meant to be exposed, however for local development minikub
e allows you to access this !
🏃  Starting tunnel for service argocd-server.
|-----------|---------------|-------------|------------------------|
| NAMESPACE |     NAME      | TARGET PORT |          URL           |
|-----------|---------------|-------------|------------------------|
| argocd    | argocd-server |             | http://127.0.0.1:39269 |
|           |               |             | http://127.0.0.1:37091 |
|-----------|---------------|-------------|------------------------|
[argocd argocd-server  http://127.0.0.1:39269
http://127.0.0.1:37091]
❗  Because you are using a Docker driver on linux, the terminal needs to be open to run it.
```

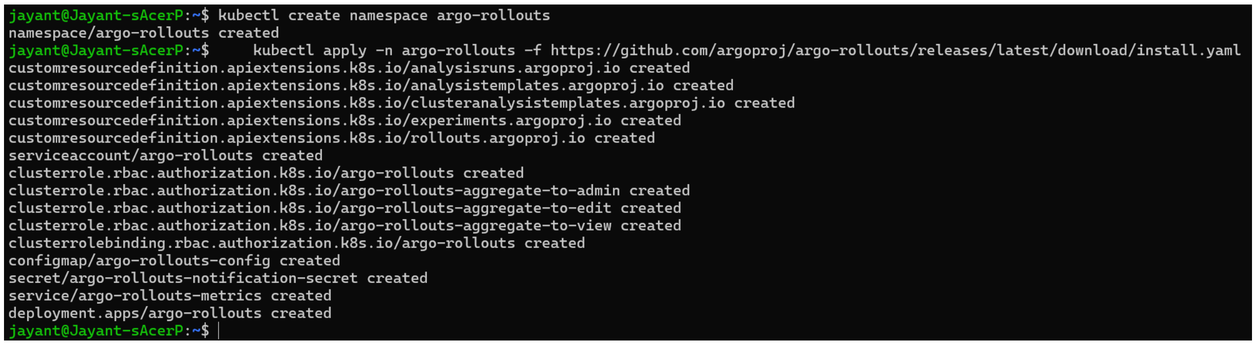


```
jayant@Jayant-sAcerP:~$ brew install argocd
==> Auto-updating Homebrew...
Adjust how often this is run with HOMEBREW_AUTO_UPDATE_SECS or disable with
HOMEBREW_NO_AUTO_UPDATE. Hide these hints with HOMEBREW_NO_ENV_HINTS (see `man brew`).
==> Downloading https://ghcr.io/v2/homebrew/core/argocd/manifests/2.10.7
################################################################################################################################################ 100.0%
==> Fetching argocd
==> Downloading https://ghcr.io/v2/homebrew/core/argocd/blobs/sha256:d21f7b8c51576892b68c96f55e9973872ac9a58796db2392d3ffa5484632f1e4
################################################################################################################################################ 100.0%
==> Pouring argocd--2.10.7.x86_64_linux.bottle.tar.gz
==> Caveats
Bash completion has been installed to:
  /home/linuxbrew/.linuxbrew/etc/bash_completion.d
==> Summary
🍺  /home/linuxbrew/.linuxbrew/Cellar/argocd/2.10.7: 8 files, 157.2MB
==> Running `brew cleanup argocd`...
Disable this behaviour by setting HOMEBREW_NO_INSTALL_CLEANUP.
Hide these hints with HOMEBREW_NO_ENV_HINTS (see `man brew`).
jayant@Jayant-sAcerP:~$ 
```

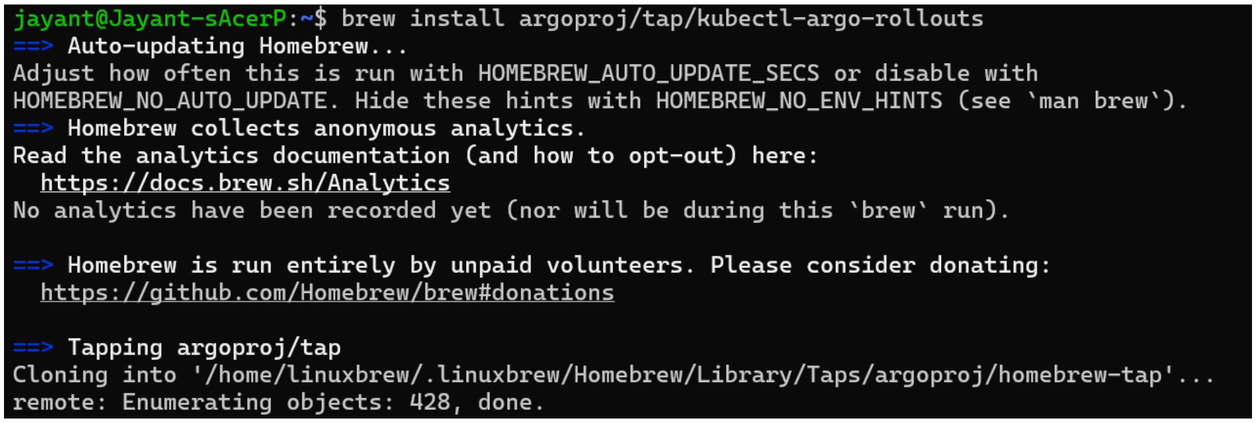
Step 4: Install Argo Rollouts

1. Argo Rollouts is a Kubernetes controller and set of CRDs that provide advanced deployment capabilities such as blue-green and canary deployments.
2. Follow the official Argo Rollouts installation guide to install it on your Kubernetes cluster.
3. This typically involves creating a namespace and installing the Argo Rollouts controller and CRDs.

Installing Argo Rollouts



```
jayant@Jayant-sAcerP:~$ kubectl create namespace argo-rollouts
namespace/argo-rollouts created
jayant@Jayant-sAcerP:~$     kubectl apply -n argo-rollouts -f https://github.com/argoproj/argo-rollouts/releases/latest/download/install.yaml
customresourcedefinition.apiextensions.k8s.io/analysisruns.argoproj.io created
customresourcedefinition.apiextensions.k8s.io/analysistemplates.argoproj.io created
customresourcedefinition.apiextensions.k8s.io/clusteranalysistemplates.argoproj.io created
customresourcedefinition.apiextensions.k8s.io/experiments.argoproj.io created
customresourcedefinition.apiextensions.k8s.io/rollouts.argoproj.io created
serviceaccount/argo-rollouts created
clusterrole.rbac.authorization.k8s.io/argo-rollouts created
clusterrole.rbac.authorization.k8s.io/argo-rollouts-aggregate-to-admin created
clusterrole.rbac.authorization.k8s.io/argo-rollouts-aggregate-to-edit created
clusterrole.rbac.authorization.k8s.io/argo-rollouts-aggregate-to-view created
clusterrolebinding.rbac.authorization.k8s.io/argo-rollouts created
configmap/argo-rollouts-config created
secret/argo-rollouts-notification-secret created
service/argo-rollouts-metrics created
deployment.apps/argo-rollouts created
jayant@Jayant-sAcerP:~$ 
```



```
jayant@Jayant-sAcerP:~$ brew install argoproj/tap/kubectl-argo-rollouts
==> Auto-updating Homebrew...
Adjust how often this is run with HOMEBREW_AUTO_UPDATE_SECS or disable with
HOMEBREW_NO_AUTO_UPDATE. Hide these hints with HOMEBREW_NO_ENV_HINTS (see `man brew`).
==> Homebrew collects anonymous analytics.
Read the analytics documentation (and how to opt-out) here:
  https://docs.brew.sh/Analytics
No analytics have been recorded yet (nor will be during this `brew` run).

==> Homebrew is run entirely by unpaid volunteers. Please consider donating:
  https://github.com/Homebrew/brew#donations

==> Tapping argoproj/tap
Cloning into '/home/linuxbrew/.linuxbrew/Homebrew/Library/Taps/argoproj/homebrew-tap'...
remote: Enumerating objects: 428, done.
```

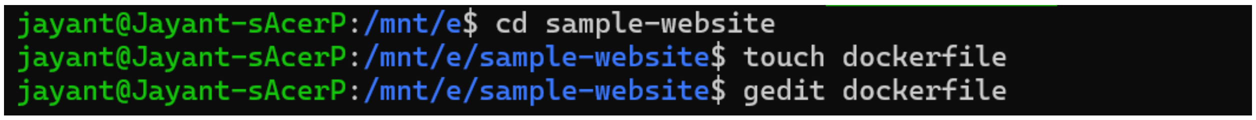
Task 2: Creating the GitOps Pipeline

Step 1: Dockerize the Application

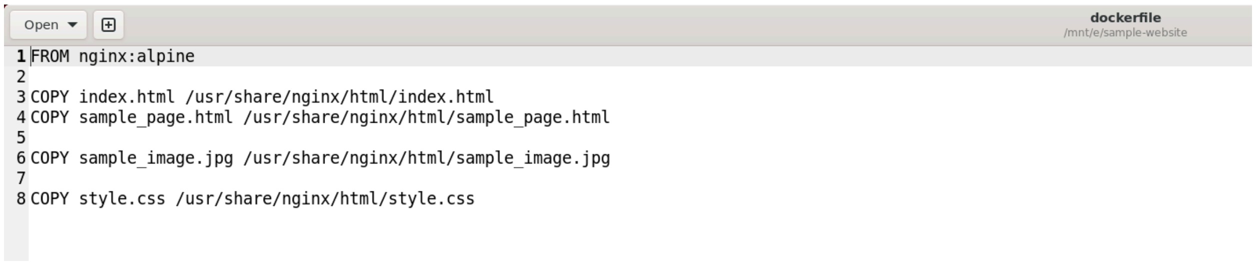
1. Choose a simple web application you want to deploy. You can use a pre-existing application or create a new one.
2. Create a `Dockerfile` in your repository to define the application's Docker image.
3. Build the Docker image locally using the `docker build` command.
4. Push the Docker image to a public container registry of your choice (e.g., Docker Hub, Google Container Registry, or Amazon Elastic Container Registry).

Dockerizing the website

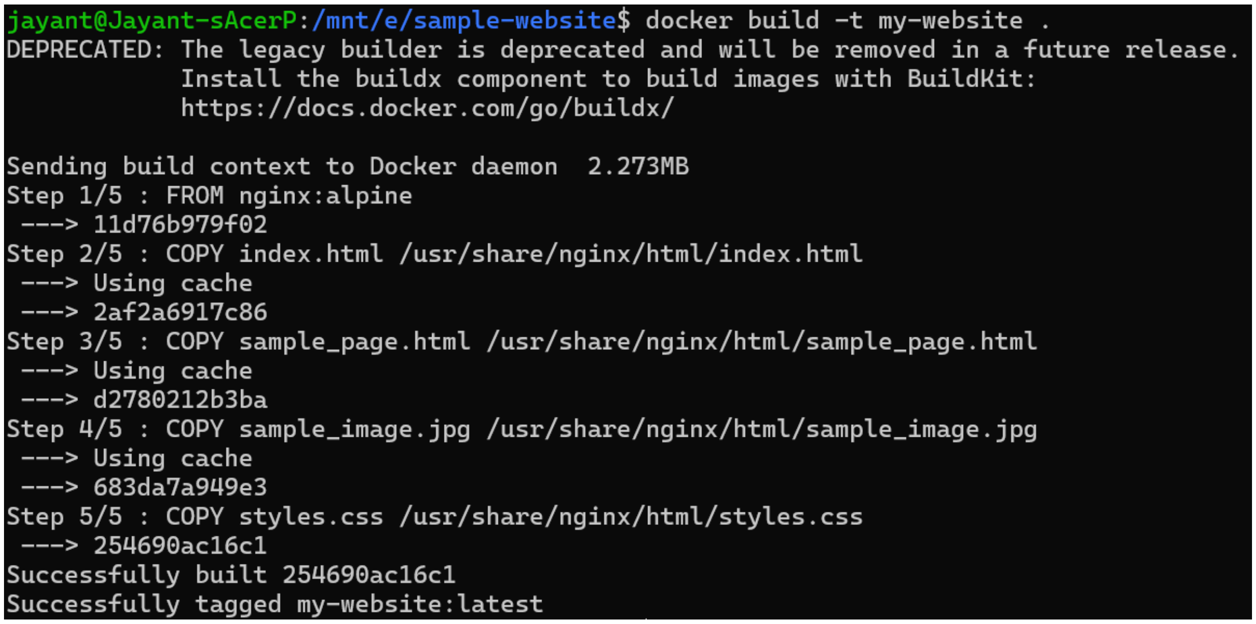
The website directory contains 4 files: index.html, sample_page.html, sample_image.jpg, style.css



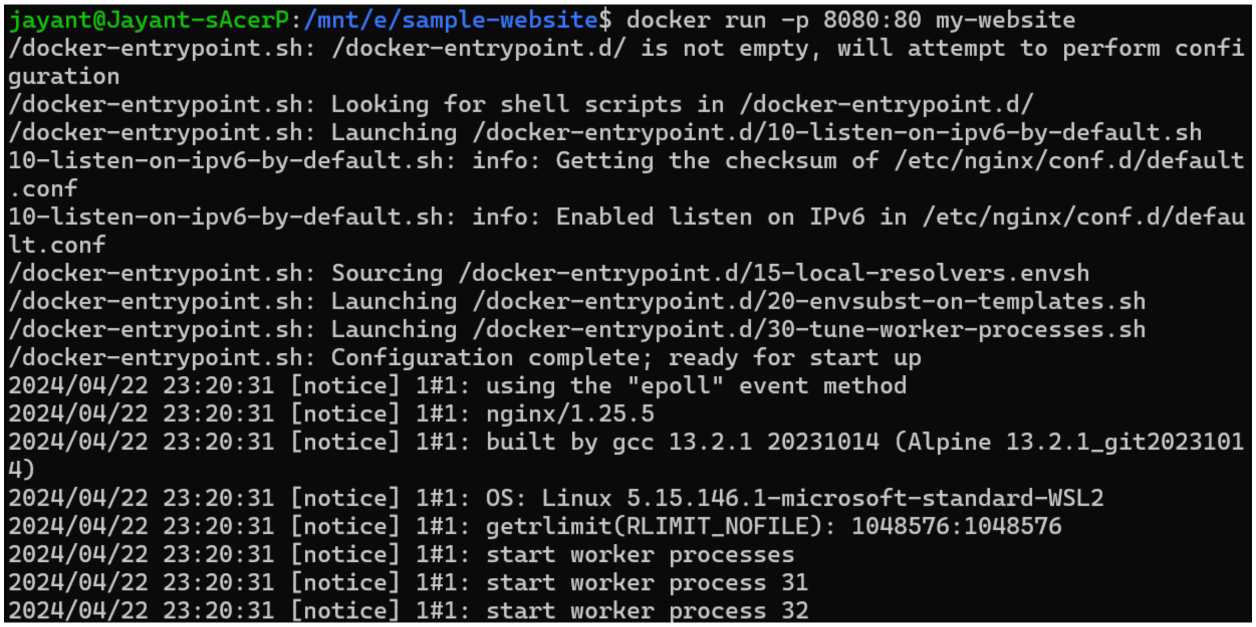
```
jayant@Jayant-sAcerP:/mnt/e$ cd sample-website
jayant@Jayant-sAcerP:/mnt/e/sample-website$ touch dockerfile
jayant@Jayant-sAcerP:/mnt/e/sample-website$ gedit dockerfile
```



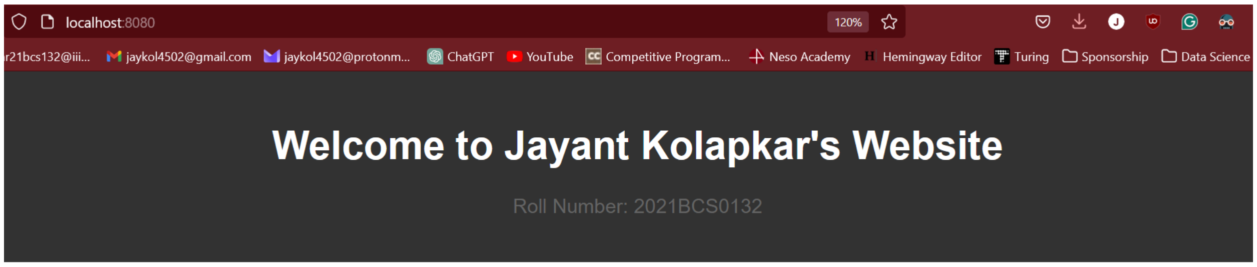
```
FROM nginx:alpine

COPY index.html /usr/share/nginx/html/index.html
COPY sample_page.html /usr/share/nginx/html/sample_page.html

COPY sample_image.jpg /usr/share/nginx/html/sample_image.jpg

COPY style.css /usr/share/nginx/html/style.css
```



```
jayant@Jayant-sAcerP:/mnt/e/sample-website$ docker build -t my-website .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  2.273MB
Step 1/5 : FROM nginx:alpine
 ---> 11d76b979f02
Step 2/5 : COPY index.html /usr/share/nginx/html/index.html
 ---> Using cache
 ---> 2af2a6917c86
Step 3/5 : COPY sample_page.html /usr/share/nginx/html/sample_page.html
 ---> Using cache
 ---> d2780212b3ba
Step 4/5 : COPY sample_image.jpg /usr/share/nginx/html/sample_image.jpg
 ---> Using cache
 ---> 683da7a949e3
Step 5/5 : COPY styles.css /usr/share/nginx/html/styles.css
 ---> 254690ac16c1
Successfully built 254690ac16c1
Successfully tagged my-website:latest
```



```
jayant@Jayant-sAcerP:/mnt/e/sample-website$ docker run -p 8080:80 my-website
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform confi
guration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default
.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/defau
lt.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2024/04/22 23:20:31 [notice] 1#1: using the "epoll" event method
2024/04/22 23:20:31 [notice] 1#1: nginx/1.25.5
2024/04/22 23:20:31 [notice] 1#1: built by gcc 13.2.1 20231014 (Alpine 13.2.1_git2023101
4)
2024/04/22 23:20:31 [notice] 1#1: OS: Linux 5.15.146.1-microsoft-standard-WSL2
2024/04/22 23:20:31 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2024/04/22 23:20:31 [notice] 1#1: start worker processes
2024/04/22 23:20:31 [notice] 1#1: start worker process 31
2024/04/22 23:20:31 [notice] 1#1: start worker process 32
```

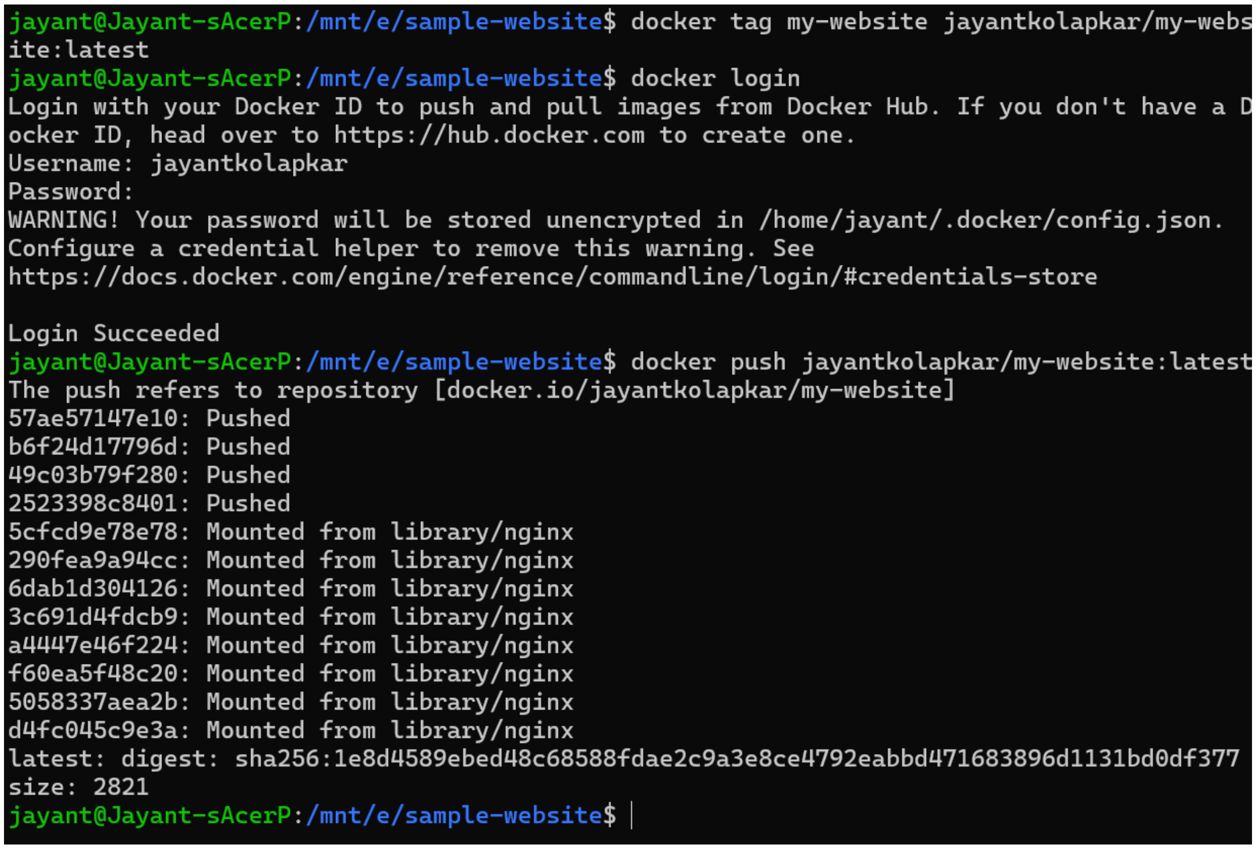


Welcome to Jayant Kolapkar's Website

Roll Number: 2021BCS0132

About Me

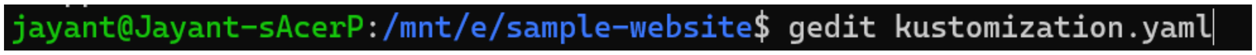
I am Jayant Kolapkar, a student pursuing B.Tech in Computer Science.



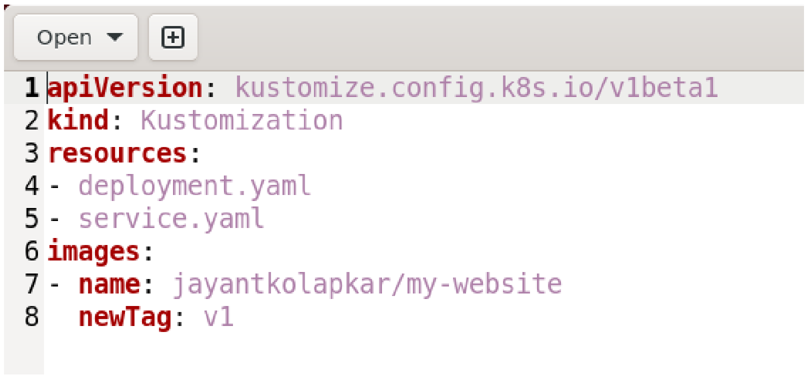
```
jayant@Jayant-sAcerP:/mnt/e/sample-website$ docker tag my-website jayantkolapkar/my-webs
ite:latest
jayant@Jayant-sAcerP:/mnt/e/sample-website$ docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a D
ocker ID, head over to https://hub.docker.com to create one.
Username: jayantkolapkar
Password:
WARNING! Your password will be stored unencrypted in /home/jayant/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
jayant@Jayant-sAcerP:/mnt/e/sample-website$ docker push jayantkolapkar/my-website:latest
The push refers to repository [docker.io/jayantkolapkar/my-website]
57ae57147e10: Pushed
b6f24d17796d: Pushed
49c03b79f280: Pushed
2523398c8401: Pushed
5cfcd9e78e78: Mounted from library/nginx
290fea9a94cc: Mounted from library/nginx
6dab1d304126: Mounted from library/nginx
3c691d4fdcb9: Mounted from library/nginx
a4447e46f224: Mounted from library/nginx
f60ea5f48c20: Mounted from library/nginx
5058337aea2b: Mounted from library/nginx
d4fc045c9e3a: Mounted from library/nginx
latest: digest: sha256:1e8d4589ebed48c68588fdae2c9a3e8ce4792eabbd471683896d1131bd0df377
size: 2821
jayant@Jayant-sAcerP:/mnt/e/sample-website$
```

Step 2: Deploy the Application Using Argo CD

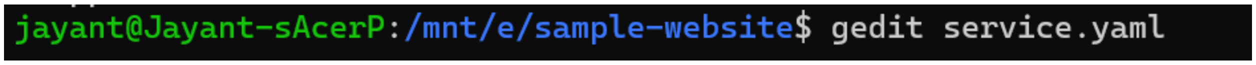
1. In your repository, create Kubernetes manifests (e.g., Deployment, Service) for your web application, using the Docker image you pushed in the previous step.
2. Configure Argo CD to monitor your repository by creating an `Application` resource in Argo CD. This resource specifies the repository URL, target Kubernetes namespace, and path to your manifests.
3. Argo CD will automatically deploy your application to the specified namespace based on the manifests in your repository.
4. Verify that your application is running in the Kubernetes cluster.



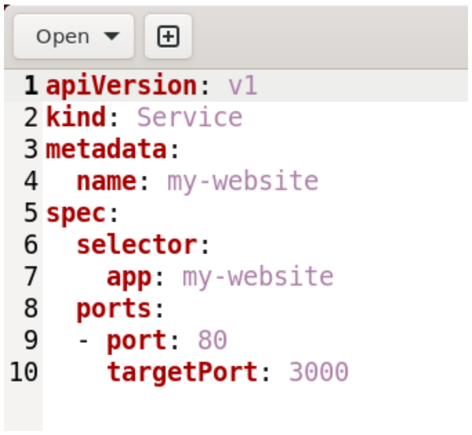
```
jayant@Jayant-sAcerP:/mnt/e/sample-website$ gedit kustomization.yaml
```



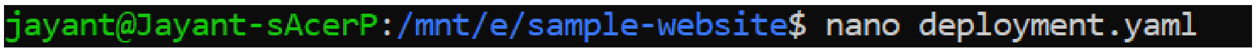
```yaml
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization
resources:
- deployment.yaml
- service.yaml
images:
- name: jayantkolapkar/my-website
  newTag: v1
```



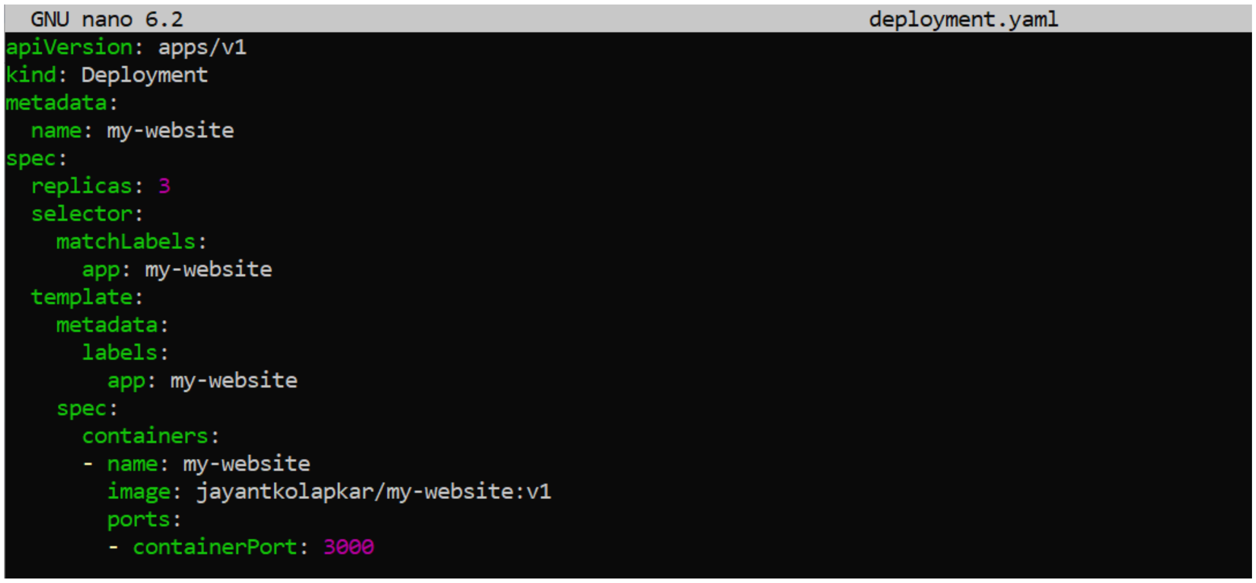
```
jayant@Jayant-sAcerP:/mnt/e/sample-website$ gedit service.yaml
```



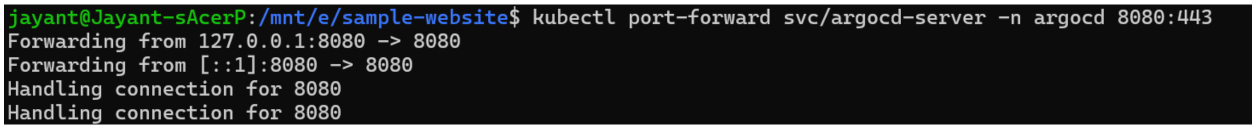
```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-website
spec:
  selector:
    app: my-website
  ports:
  - port: 80
    targetPort: 3000
```



```
jayant@Jayant-sAcerP:/mnt/e/sample-website$ nano deployment.yaml
```



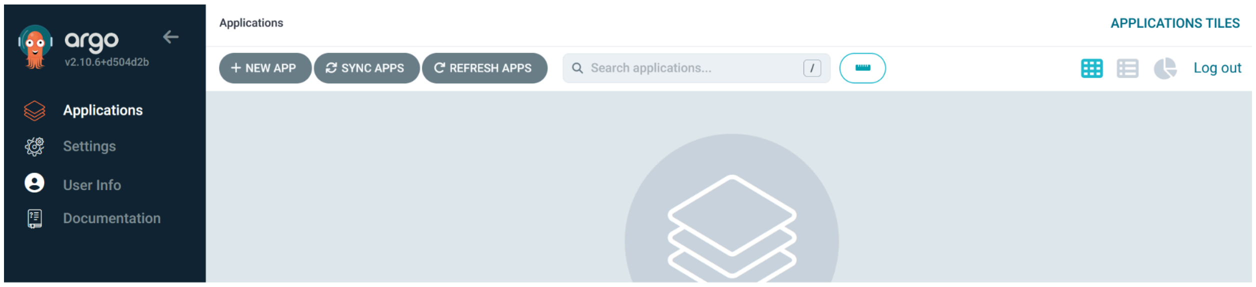
```
  GNU nano 6.2                                        deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-website
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-website
  template:
    metadata:
      labels:
        app: my-website
    spec:
      containers:
      - name: my-website
        image: jayantkolapkar/my-website:v1
        ports:
        - containerPort: 3000
```

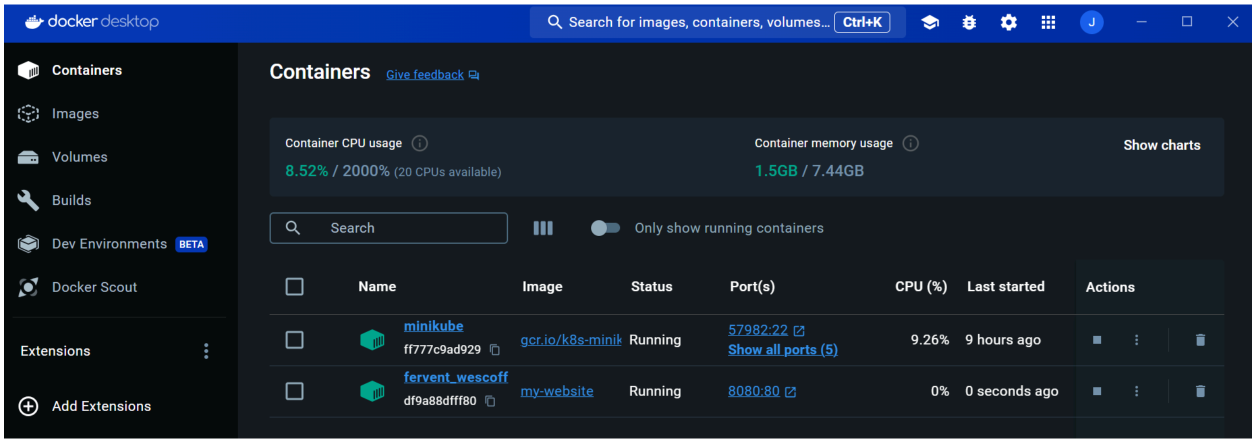


```
jayant@Jayant-sAcerP:/mnt/e/sample-website$ kubectl port-forward svc/argocd-server -n argocd 8080:443
Forwarding from 127.0.0.1:8080 -> 8080
Forwarding from [::1]:8080 -> 8080
Handling connection for 8080
Handling connection for 8080
```

'admin' is the username. To get the one time password for logging in, run:

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



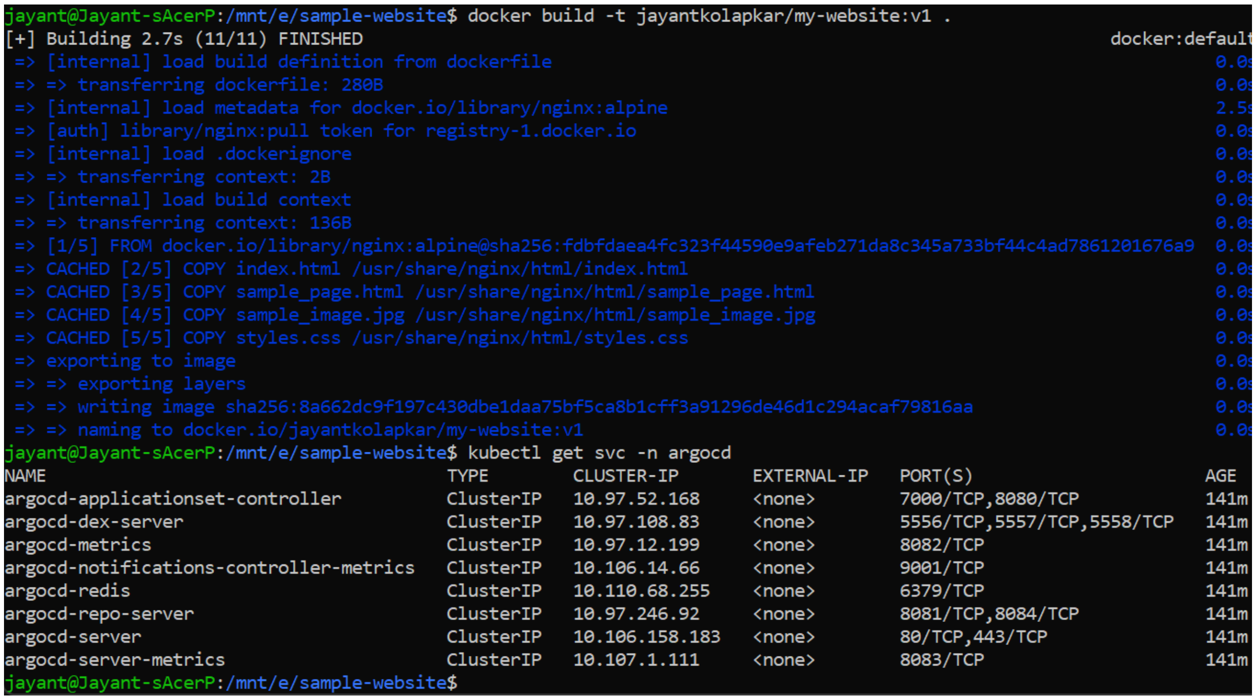


Containers

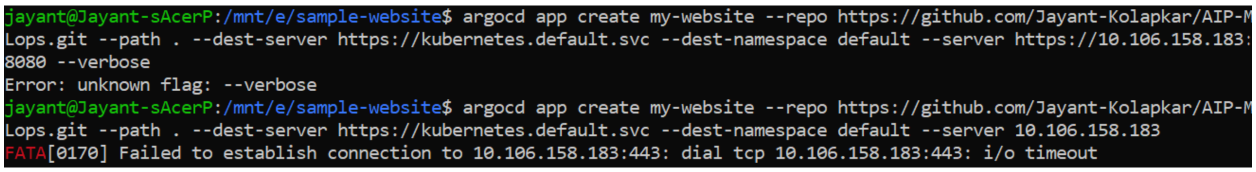
Container CPU usage
8.52% / 2000% (20 CPUs available)

Container memory usage
1.5GB / 7.44GB

Name	Image	Status	Port(s)	CPU (%)	Last started	Actions
minikube ff777c9ad929	gcr.io/k8s-minik	Running	57982:22 Show all ports (5)	9.26%	9 hours ago	
fervent_wescoff df9a88dfff80	my-website	Running	8080:80	0%	0 seconds ago	



```
jayant@Jayant-sAcerP:/mnt/e/sample-website$ docker build -t jayantkolapkar/my-website:v1 .
[+] Building 2.7s (11/11) FINISHED                                                  docker:default
 => [internal] load build definition from dockerfile                                          0.0
 => => transferring dockerfile: 280B                                                          0.0
 => [internal] load metadata for docker.io/library/nginx:alpine                               2.5
 => [auth] library/nginx:pull token for registry-1.docker.io                                  0.0
 => [internal] load .dockerignore                                                             0.0
 => => transferring context: 2B                                                               0.0
 => [internal] load build context                                                             0.0
 => => transferring context: 136B                                                             0.0
 => [1/5] FROM docker.io/library/nginx:alpine@sha256:fdbfdaea4fc323f44590e9afeb271da8c345a733bf44c4ad7861201676a9  0.0
 => CACHED [2/5] COPY index.html /usr/share/nginx/html/index.html                             0.0
 => CACHED [3/5] COPY sample_page.html /usr/share/nginx/html/sample_page.html                 0.0
 => CACHED [4/5] COPY sample_image.jpg /usr/share/nginx/html/sample_image.jpg                 0.0
 => CACHED [5/5] COPY styles.css /usr/share/nginx/html/styles.css                             0.0
 => exporting to image                                                                        0.0
 => => exporting layers                                                                       0.0
 => => writing image sha256:8a662dc9f197c430dbe1daa75bf5ca8b1cff3a91296de46d1c294acaf79816aa  0.0
 => => naming to docker.io/jayantkolapkar/my-website:v1                                       0.0
jayant@Jayant-sAcerP:/mnt/e/sample-website$ kubectl get svc -n argocd
NAME                                      TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)                      AGE
argocd-applicationset-controller          ClusterIP   10.97.52.168     <none>        7000/TCP,8080/TCP            141m
argocd-dex-server                         ClusterIP   10.97.108.83     <none>        5556/TCP,5557/TCP,5558/TCP   141m
argocd-metrics                            ClusterIP   10.97.12.199     <none>        8082/TCP                     141m
argocd-notifications-controller-metrics   ClusterIP   10.106.14.66     <none>        9001/TCP                     141m
argocd-redis                              ClusterIP   10.110.68.255    <none>        6379/TCP                     141m
argocd-repo-server                        ClusterIP   10.97.246.92     <none>        8081/TCP,8084/TCP            141m
argocd-server                             ClusterIP   10.106.158.183   <none>        80/TCP,443/TCP               141m
argocd-server-metrics                     ClusterIP   10.107.1.111     <none>        8083/TCP                     141m
jayant@Jayant-sAcerP:/mnt/e/sample-website$
```



```
jayant@Jayant-sAcerP:/mnt/e/sample-website$ argocd app create my-website --repo https://github.com/Jayant-Kolapkar/AIP-M
Lops.git --path . --dest-server https://kubernetes.default.svc --dest-namespace default --server https://10.106.158.183:
8080 --verbose
Error: unknown flag: --verbose
jayant@Jayant-sAcerP:/mnt/e/sample-website$ argocd app create my-website --repo https://github.com/Jayant-Kolapkar/AIP-M
Lops.git --path . --dest-server https://kubernetes.default.svc --dest-namespace default --server 10.106.158.183
FATA[0170] Failed to establish connection to 10.106.158.183:443: dial tcp 10.106.158.183:443: i/o timeout
```

Here is where I encountered the unsolvable barrier. ArgoCD would show me the running IP of the server, but would not let me connect to it no matter what. I tried to replicate the same on Windows, but was faced with the same problem. So I directly uploaded the whole website directory to the GitHub Repository. The next are just steps in detail.

Task 3: Implementing a Canary Release with Argo Rollouts

Canary Releases

A canary release is a deployment strategy where a new version of an application is gradually rolled out to a small subset of users or traffic before being fully deployed to the entire user base. This approach allows you to test the new version in a production environment while minimizing the risk of introducing bugs or issues that could impact all users.

The term "canary" refers to the historical practice of using canaries in coal mines to detect toxic gases. If the canary became ill or died, it would signal danger and alert the miners to evacuate. Similarly, in a canary release, the initial subset of users acts as the "canary," allowing you to detect and address any issues before rolling out the new version more broadly.

Argo Rollouts

Argo Rollouts is a Kubernetes controller and set of custom resource definitions (CRDs) that provide advanced deployment capabilities, including canary releases, blue-green deployments, and other progressive delivery strategies.

With Argo Rollouts, you can define the desired deployment strategy for your application in a declarative manner, and the controller will orchestrate the deployment process according to your specifications. This allows you to automate and control the rollout process, including steps like traffic routing, scaling, and analysis of metrics and logs.

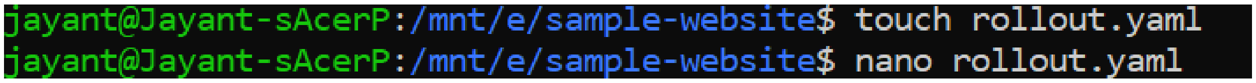
Deployment Strategies

Argo Rollouts supports several deployment strategies, including:

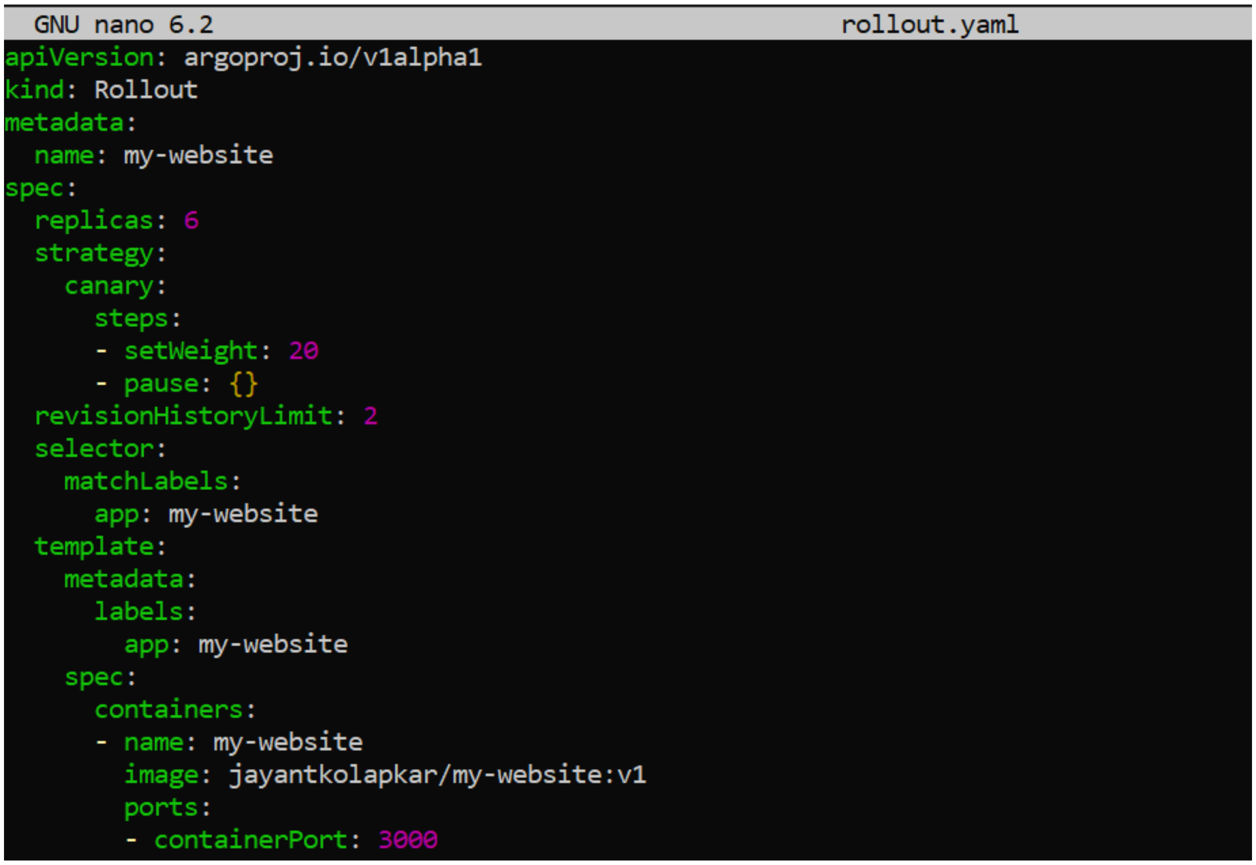
1. **Canary**: A new version is gradually rolled out to a specified percentage of traffic, allowing you to test it in production before fully deploying it.
2. **Blue-Green**: Two separate environments (blue and green) are maintained, and traffic is shifted between them during a deployment.
3. **Rolling Update**: The new version is gradually rolled out to replace the old version, with a configurable percentage of instances being updated at a time.
4. **A/B Testing**: Two versions of the application are deployed simultaneously, and traffic is split between them based on defined criteria, allowing you to test different versions with real users.

Step 1: Define a Rollout Strategy

1. Modify your application's Deployment manifest to use the Argo Rollouts `Rollout` resource instead of a standard Deployment.
2. In the `Rollout` definition, specify a canary release strategy by setting the `strategy.canary` field. This defines the percentage of traffic that should be routed to the new version during the canary phase.



```
jayant@Jayant-sAcerP:/mnt/e/sample-website$ touch rollout.yaml
jayant@Jayant-sAcerP:/mnt/e/sample-website$ nano rollout.yaml
```

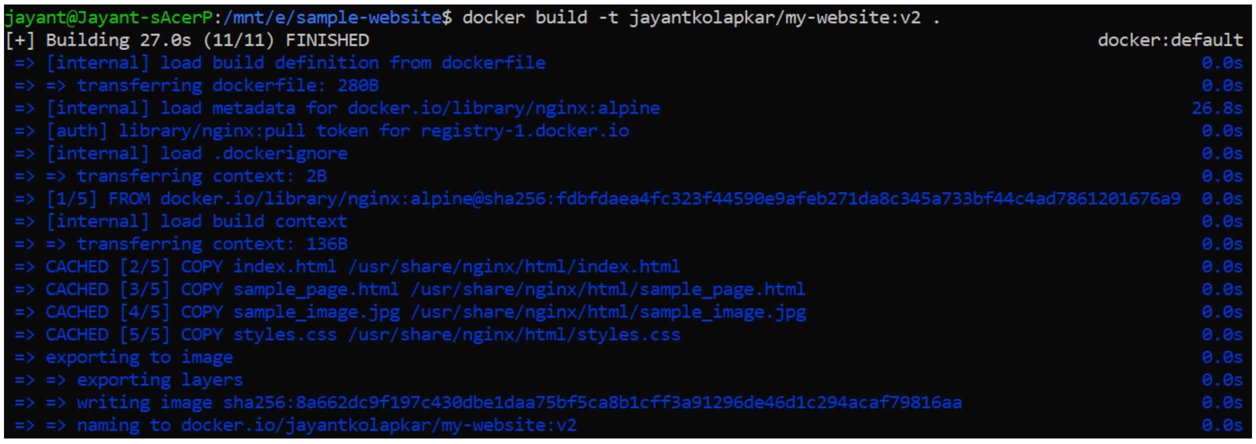


```
  GNU nano 6.2                                            rollout.yaml
apiVersion: argoproj.io/v1alpha1
kind: Rollout
metadata:
  name: my-website
spec:
  replicas: 6
  strategy:
    canary:
      steps:
      - setWeight: 20
      - pause: {}
  revisionHistoryLimit: 2
  selector:
    matchLabels:
      app: my-website
  template:
    metadata:
      labels:
        app: my-website
    spec:
      containers:
      - name: my-website
        image: jayantkolapkar/my-website:v1
        ports:
        - containerPort: 3000
```

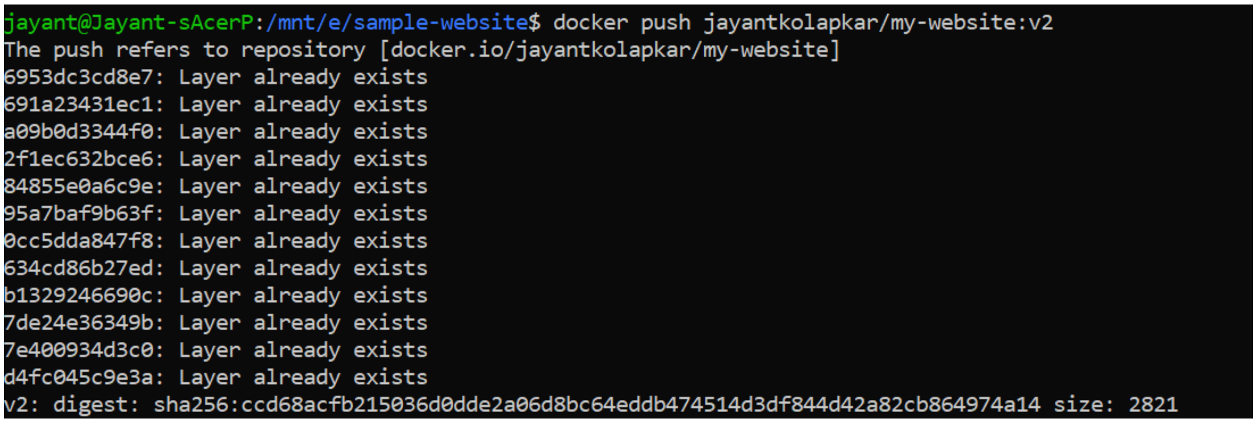
Step 2: Trigger a Rollout

1. Make a change to your web application's code and rebuild the Docker image with a new tag.
2. Push the new Docker image to your container registry.
3. Update the `Rollout` manifest in your repository to use the new Docker image tag.

4. Argo CD will detect the changes in your repository and trigger a new rollout, following the canary release strategy you defined.



```
jayant@Jayant-sAcerP:/mnt/e/sample-website$ docker build -t jayantkolapkar/my-website:v2 .
[+] Building 27.0s (11/11) FINISHED                                                            docker:default
 => [internal] load build definition from dockerfile                                                     0.0s
 => => transferring dockerfile: 280B                                                                     0.0s
 => [internal] load metadata for docker.io/library/nginx:alpine                                         26.8s
 => [auth] library/nginx:pull token for registry-1.docker.io                                             0.0s
 => [internal] load .dockerignore                                                                        0.0s
 => => transferring context: 2B                                                                          0.0s
 => [1/5] FROM docker.io/library/nginx:alpine@sha256:fdbfdaea4fc323f44590e9afeb271da8c345a733bf44c4ad7861201676a9  0.0s
 => [internal] load build context                                                                        0.0s
 => => transferring context: 136B                                                                        0.0s
 => CACHED [2/5] COPY index.html /usr/share/nginx/html/index.html                                        0.0s
 => CACHED [3/5] COPY sample_page.html /usr/share/nginx/html/sample_page.html                            0.0s
 => CACHED [4/5] COPY sample_image.jpg /usr/share/nginx/html/sample_image.jpg                            0.0s
 => CACHED [5/5] COPY styles.css /usr/share/nginx/html/styles.css                                        0.0s
 => exporting to image                                                                                   0.0s
 => => exporting layers                                                                                  0.0s
 => => writing image sha256:8a662dc9f197c430dbe1daa75bf5ca8b1cff3a91296de46d1c294acaf79816aa             0.0s
 => => naming to docker.io/jayantkolapkar/my-website:v2                                                  0.0s
```



```
jayant@Jayant-sAcerP:/mnt/e/sample-website$ docker push jayantkolapkar/my-website:v2
The push refers to repository [docker.io/jayantkolapkar/my-website]
6953dc3cd8e7: Layer already exists
691a23431ec1: Layer already exists
a09b0d3344f0: Layer already exists
2f1ec632bce6: Layer already exists
84855e0a6c9e: Layer already exists
95a7baf9b63f: Layer already exists
0cc5dda847f8: Layer already exists
634cd86b27ed: Layer already exists
b1329246690c: Layer already exists
7de24e36349b: Layer already exists
7e400934d3c0: Layer already exists
d4fc045c9e3a: Layer already exists
v2: digest: sha256:ccd68acfb215036d0dde2a06d8bc64eddb474514d3df844d42a82cb864974a14 size: 2821
```

Step 3: Monitor the Rollout

1. Use the Argo Rollouts UI or CLI to monitor the progress of the canary release.
2. Argo Rollouts will gradually shift traffic to the new version, allowing you to validate its behavior before fully promoting it.
3. If the canary release is successful, you can promote the new version to serve 100% of the traffic. If issues arise, you can abort the rollout and roll back to the previous stable version.
4. If the new version performs well, you can resume the Rollout by clicking the "RESUME" button in the Argo CD UI or using the Argo Rollouts CLI: `kubectl argo rollouts promote my-website`
5. If issues are detected with the new version, you can abort the Rollout and roll back to the previous stable version using the Argo CD UI or the Argo Rollouts CLI: `kubectl argo rollouts abort my-website`

Task 4: Documentation and Cleanup

Step 1: Document the Process

1. Create a `README.md` file in your repository, detailing the steps you followed for this assignment.
2. Include any challenges you encountered and how you resolved them.
3. Explain the GitOps principles you followed and how Argo CD and Argo Rollouts facilitated the deployment and release process.

Step 2: Clean Up

1. In the `README.md` file, document the steps to cleanly remove all resources created during this assignment from your Kubernetes cluster.
2. Delete the my-website application from Argo CD: `argocd app delete my-website`
3. This may include deleting the Argo CD and Argo Rollouts installations, removing the deployed application resources, and any other related resources you created:

```
kubectl delete -n argocd -f
https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml

kubectl delete -n your-namespace -f
https://raw.githubusercontent.com/argoproj/argo-rollouts/master/manifests/install.yaml
```

4. Stop Minikube instances by:

```
minikube stop

minikube delete
```