PHASE 1 DevOps Infrastructure Project

Wednesday, 17 September 2025 6:43 am

Goal: To understand how to build and deploy modern cloud-native infrastructure

Key Components:

- VM setup with Vmware
- Kubernetes (K3s cluster setup)
- CNI plugins (Calico)
- Dockerfile + app manifests (sampapp, DB, Redis, etc)
- GitHub Project structure
- Network Policies + DNS Troubleshooting
- Persistent Volumes + StorageClass
- Cronjob for postgres backups
- Ingress Setup via Helm
- CI/CD: GitHub > Jenkins > ArgoCD (GitOps)
- Ansible for node provisioning
- Istio ServiceMesh
- Prometheus + Grafana + Kiali
- Middleware, MessageBroker: RabbitMQ

Designed and deployed Kubernetes-based DevOps lab from scratch with Calico CNI, Helm-based Ingress, and GitOps workflow via Jenkins and ArgoCD

- **WHAT I DID**
  - **VM set up (Vmware)**
  - **SSH set up between my host machine and the 2 VMs**
  - **Installed Docker on both VMs**
  - **Installed K3s on both VMs and joined worker node to master node cluster**
  - **Set up cluster access from local (kube config)**
  - **Created simple feedback form app**
  - **Created a Dockerfile for app**
  - **Built and Pushed image to DockerHub**
  - **Created pod yaml manifests**
  - **Pushed yamls to GitHub**
  - **Deployed YAMLs to cluster**
  - **Replaced Flannel with Calico**
  - **Set up some test Netpols**
  - **Created PVC for Postgres**
  - **Cronjob for backups for Postgres**
  - **GitHub Resync (as some edits were made to cluster directly)**
  - **Ingress**
  - **CI/CD with Jenkins & ArgoCD**
  - **Node Provisioning with Ansible**
  - **Istio ServiceMesh**
  - **Prometheus + Grafana + Kiali for observability**
  - **RabbitMQ**

Each section will show the steps I carried out, and if any issues were faced, a short description of the ISSUE is included, with a reference to which Troubleshooting Log page to refer to for more detailed explanations (the issue number within the reference Troubleshooting Log page will be the same as in

the section). But do note that the issues may not be in chronological order - pls follow Reference Log # AND Issue #.

Troubleshooting Log 1 - Miscellaneous Topics Issues
Troubleshooting Log 2 - Postgres
Troubleshooting Log 3 - Ingress
Troubleshooting Log 4 - CI/CD w Jenkins and ArgoCD

1 VM and ssh setup

Sunday, 10 August 2025 8:08 pm

Technology:
Vmware Workstation Pro 17

VM Creation

1. Click on "Create a New Virtual Machine"
2. Select Installer disc image file (iso)
   - Click on Browse and navigate to the directory I saved the ISO Image:
   - *C:\Users\mikia\Downloads\ubuntu-22.04.5-live-server-amd64.iso*
3. Name VM (master1 and worker1)
4. Select "Store virtual disk as a single file"
5. Customize Hardware > New CD/DVD (SATA) change to Auto detect
6. Click Finish and let OS boot up

System Bootup

1. Click on first option in GRUB menu (Try or install Ubuntu Server)
2. Keep clicking Done for all the options offered
3. Name the server as your name, nodename, your username and pw
4. Select Install OpenSSH server
5. Wait for configurations to complete then click Reboot Now
6. When screen shows "failed to mount /cdrom", press enter
7. Login to server and shutdown now
8. Click on VM settings and edit CD/DVD (SATA) to Use physical drive (autodetect)
9. Power on VM again

SSH Setup

1. **On host machine, generate the ssh id_rsa private and public keys:**

```
$ ssh-keygen -t rsa -b 4096 -C "WorkerNode"
```

-t = key type
    Rsa is RSA key pair… alt include ed25519, ecdsa etc
-b = bits
    For RSA, common values are 2048 or 4096
    4096 is stronger encryption but a little slower than 2048
    For ed25519, don't need -b because it's fixed strength
-C = comment only
We will use same key for both master1 and worker1 nodes.

2. **Copy contents of the id_rsa.pub key and copy over to /home/miki/.ssh/authorized_keys in VM**

```
$ printf '%s\n' 'ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAACAQCx3A54D3j0mVcDNajgY/U3RI1aGN20vPm8/VVLsStUOfjX
y1Kt0t4z0WV+RLjAmKUM4DH9/VKzOlue/XYTJQS35pUGBC/lcEfCHTTN3INdMcYctps5OSyIEhIN/A5
ramlxTMy25qH5c860lfcwoxnd52ksmly+efmtpMaizUKFhiH28T3MTKm+lT3mqo8XvTjjYnX9XRujF5/
FLSm4ChCpbBu8vEUa+tVjQT6MmAKrypdcqd0g3a79xr3wUq6pzLLM/VMsMkvt3c8lw9QV6rmbvEY
n7UNQt4hgF3r+Exz++qkX00Por/S7971QCTYi/EY3Xdy8MVnoXLIacDabWwurtcn8JQ6DGZDd2HBI4
oeLfK/e258oJ6BWVT/YDMdcv7ApRGjLc7d7OT1Se74hnPoEec1lh7eeOsVmVaKzh0o2rTMfBoffCmP
odikhB+bTirWS0v3JwgDill6wXO8ZqNpAkREw/jqGh0tC7lb2qZg55LGu/z2g0vRi7TJuFP9O0AOLhZ+q
i6DOPVNz9FTUU40c/qzTvbyvLhKEFMlSqnXOZ9FuIlfpp6CxfZYkWL4OKXS1uLE9jrbPxrM6d4oy3549
Vyv3Anq4+IVMQUH6oVAlO9lztVjd3B+QMpRMsAmnEN9CpVE+HXBE2+
```

```
7JO8P3gwNdNbK6Z5Nfsj/ZfUUqu5qERw== ubuntu-key' > /home/miki/.ssh/authorized_keys
```

3. **Ensure key is a one liner -- :**

```
$ sed -i 's/\r$//' /home/miki/.ssh/authorized_keys
```

4. **Change ownership of .ssh dir and contents :**

```
$ chown -R miki:miki /home/miki/.ssh/
```

5. **Change permissions**

```
$ chmod 600 /home/miki/.ssh/authorized_keys
$ chmod 700 /home/miki/.ssh/
```

6. **Restart ssh service**

```
$ systemctl restart ssh
```

7. **Verify effective sshd config:**

```
$ sudo sshd -T | grep -E "pubkeyauthentication|authorizedkeysfile|passwordauthentication"
```

Expected Output:

pubkeyauthentication yes
passwordauthentication yes
authorizedkeysfile .ssh/authorized_keys .ssh/authorized_keys2

8. **Remove # from AuthorizedKeysFile line in sshd_config**

```
# sudo grep -i authorized_keys /etc/ssh/sshd_config
[sudo] password for miki:
# Expect .ssh/authorized_keys2 to be disregarded by default in future.
#AuthorizedKeysFile     .ssh/authorized_keys .ssh/authorized_keys2
```

9. **To access with nodenames instead of IP**

Make a ~/.ssh/config file in local machine (can use Git Bash)

```
# Master node
Host master1
        HostName 192.168.81.128
        User miki
        IdentityFile ~/.ssh/id_rsa

# Worker node 1
Host worker1
        HostName 192.168.81.129
        User miki
        IdentityFile ~/.ssh/id_rsa
```

Change permissions of file to 600. Now can ssh in to master1 and worker1

To debug ssh:

```
ssh -vvv miki@192.168.81.128 / master1
```

Had to regenerate ssh key bc git bash/windows was creating some string for my passphrase that I couldn't ssh using.

```
$ mv ~/.ssh/id_rsa ~/.ssh/id_rsa.bak

$ mv ~/.ssh/id_rsa.pub ~/.ssh/id_rsa.pub.bak

$ ssh-keygen -t rsa -b 4096 -C "localhost" -N "" -f ~/.ssh/id_rsa
Generating public/private rsa key pair.
Your identification has been saved in /c/Users/mikia/.ssh/id_rsa
Your public key has been saved in /c/Users/mikia/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:gfuPxY3HbvkoLWqfQHhWiQiTr8VHtTCIF5QFAR17+l0 localhost
The key's randomart image is:
+---[RSA 4096]----+
|    +**O+..      |
|    oo=+.+ o     |
|     ++.+ +      |
|      +=.o       |
|     o+.S   E    |
|    .  * o =     |
|       + =.+.    |
|       .=o++.    |
|      .oo++o..   |
+----[SHA256]-----+
```

Copy to node

```
$ ssh-copy-id -i ~/.ssh/id_rsa.pub miki@192.168.81.128
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/c/Users/mikia/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are
already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to
install the new keys
miki@192.168.81.128's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh -i /c/Users/mikia/.ssh/id_rsa 'miki@
192.168.81.128'"
and check to make sure that only the key(s) you wanted were added.
```

2 Install Docker

Sunday, 10 August 2025 9:11 pm

Technology:
Docker

https://docs.docker.com/engine/install/ubuntu/#install-using-the-repository

1. **Set up Docker's apt repo**

```
# Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "${UBUNTU_CODENAME:-$VERSION_CODENAME}") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```

2. **Install the Docker packages**

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

3. **Verify installation success**

```
sudo docker run hello-world
```

3 Install K3s

Sunday, 10 August 2025 9:14 pm

Technology:
K3s

On Control plane node:

1. Update your system

```
# sudo apt update && sudo apt upgrade -y
```

2. Install curl if not already installed

```
# sudo apt install -y curl
```

3. Install k3s

```
# curl -sfL https://get.k3s.io | sh -
```

4. Wait for k3s to be fully installed and running

Check the status

```
# sudo systemctl status k3s
```

5. Verify installation

```
# sudo k3s kubectl get nodes
```

6. Export kubeconfig for easy access

```
# sudo k3s kubectl config view --raw > ~/.kube/config
# export KUBECONFIG=~/.kube/config
```

7. Make a user copy of kubeconfig so I can access cluster from username miki

```
mkdir -p ~/.kube
sudo cp /etc/rancher/k3s/k3s.yaml ~/.kube/config
sudo chown $USER:$(id -gn) ~/.kube/config
chmod 600 ~/.kube/config
unset KUBECONFIG
export KUBECONFIG=$HOME/.kube/config
kubectl get no
```

8. Check cluster -- master node shld be there

```
# kubectl get no
```

8. Retrieve join token from master node

```
# cat /var/lib/rancher/k3s/server/node-token
```

On Worker Node:

1. Join worker node to cluster

```
# curl -sfL https://get.k3s.io | K3S_URL=https://<MASTER_IP>:6443 K3S_TOKEN=<NODE_TOKEN> sh -
```

2. Check service

```
# systemctl status k3s-agent
```

Back on Master Node:

1. Verify both nodes are connected

```
# k get no
```

4 Simple App creation

Monday, 11 August 2025 8:48 pm

Step 1: Install Python Tools (from here is on Vscode terminal)

Run in windows terminal (in SampApp dir):

```
# python -m venv venv
# .\venv\Scripts\activate
```

** If you get a security policy error, run:
$ Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope Process
then try again.

```
# pip install flask
```

Step 2: Project Folder Structure

Create these files:

SampApp/
- **k8s-manifests/**
  - (contents will come later)
- **static/**
  - **style.css**
- **templates**
  - index.html
  - success.html
- **feedback.db**
- **requirements.txt**
- **app.py**

static/style.css

```
body {
    font-family: Arial, sans-serif;
    background-color: #f4f4f4;
}
.container {
    width: 40%;
    margin: auto;
    background: white;
    padding: 2em;
    margin-top: 5%;
    box-shadow: 0 0 10px rgba(0,0,0,0.1);
}
input, textarea {
    width: 100%;
    padding: 0.8em;
    margin-top: 1em;
}
```

```
button {
    padding: 0.8em 2em;
    margin-top: 1em;
}
```

templates/index.html

```
<html>
<head>
    <title>Feedback Form</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>
<body>
    <div class="container">
        <h2>Feedback Form</h2>
        <form method="POST">
            <input type="text" name="name" placeholder="Your Name" required>
            <input type="email" name="email" placeholder="Your Email" required>
            <textarea name="message" placeholder="Your Feedback" required>
</textarea>
            <button type="submit">Submit</button>
        </form>
    </div>
</body>
</html>
```

templates/success.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Success</title>
</head>
<body>
    <h2>Thanks for your feedback, {{ name }}!</h2>
</body>
</html>
```

requirements.txt

```
Flask
```

app.py

```
from flask import Flask, render_template, request
import sqlite3
import os
app = Flask(__name__)
DB_PATH = 'feedback.db'
def init_db():
    if not os.path.exists(DB_PATH):
        conn = sqlite3.connect(DB_PATH)
        c = conn.cursor()
        c.execute('''CREATE TABLE feedback
                     (id INTEGER PRIMARY KEY AUTOINCREMENT,
                      name TEXT NOT NULL,
                      email TEXT NOT NULL,
                      message TEXT NOT NULL)''')
        conn.commit()
        conn.close()
```

```
@app.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        name = request.form['name']
        email = request.form['email']
        message = request.form['message']
        conn = sqlite3.connect(DB_PATH)
        c = conn.cursor()
        c.execute("INSERT INTO feedback (name, email, message) VALUES
(?, ?, ?)", (name, email, message))
        conn.commit()
        conn.close()
        return render_template('success.html', name=name)
    return render_template('index.html')
if __name__ == '__main__':
    init_db()
    app.run(host='0.0.0.0', port=5000)
```

5 Dockerfile

Wednesday, 3 September 2025 8:55 pm

Build Dockerfile in app's root dir

```
FROM python:3.10-slim
# Set env var
ENV PYTHONDONTWRITEBYTECODE=1
ENV PYTHONUNBUFFERED=1
# Set the working directory in the container
WORKDIR /app
# Copy requirements first (for caching efficiency)
COPY requirements.txt .
# Install dependencies
RUN pip install --no-cache-dir -r requirements.txt
# Copy the rest of the application
COPY . .
# Expose port (must match app.py)
EXPOSE 5000
# Run the Flask app
CMD ["python", "app.py"]
```

6 Push image to Docker Hub

Sunday, 7 September 2025 1:26 pm

Download Docker desktop and make account on Dockerhub and desktop (I use Github acc for both).

Check that docker desktop is installed, do a docker info to check that there is output.

Open docker desktop.

On vscode Terminal, go into main app dir.

1. Build image

*docker desktop mac and windows uses BuildKit by default (buildx is part of it)
*Image and repo name shld be lower case
docker buildx build --load -t username/repo_name:tag root_dir

```
docker buildx build --load -t mikiangelica/sampapp:v1 .
```

Check that image is created:
docker images

2. Push image

Make sure to login with:

```
docker login
```

Push image:

```
docker push mikiangelica/sampapp:v1
```

*image name shld match repo name basically

Can check on docker hub UI if image is there

7 Pod yaml manifests

Sunday, 7 September 2025 1:41 pm

Create the different pod manifests (app, postgres, redis, nginx, busybox)

- **Sampapp** (main app)
- **Postgres** (db)
- **Redis** (cache)
- **Nginx** (reverse proxy for webapp AKA receives incoming HTTP requests and forwards to actual app)
- **Busybox** (for testing)

k8s-manifests/
├── sampapp.yaml
├── postgres.yaml
├── redis.yaml
├── nginx.yaml
├── busybox.yaml

Sampapp

```yaml
apiVersion: v1
kind: Service
metadata:
  name: sampapp-service
spec:
  selector:
    app: sampapp
  ports:
    - protocol: TCP
      port: 5000
      targetPort: 5000
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sampapp
  labels:
    app: sampapp
spec:
  replicas: 2
  selector:
    matchLabels:
      app: sampapp
  template:
    metadata:
      labels:
        app: sampapp
    spec:
      containers:
      - name: sampapp
        image: mikiangelica/sampapp:v1
```

```
        ports:
        - containerPort: 5000
```

Postgres

```
apiVersion: v1
kind: Service
metadata:
  name: postgres-service
spec:
  selector:
    app: postgres
  ports:
    - protocol: TCP
      port: 5432
      targetPort: 5432
  ---
  apiVersion: apps/v1
kind: Deployment
metadata:
  name: postgres
  labels:
    app: postgres
spec:
  replicas: 1
  selector:
    matchLabels:
      app: postgres
  template:
    metadata:
      labels:
        app: postgres
    spec:
      containers:
      - name: postgres
        image: postgres:14
        ports:
        - containerPort: 5432
        env:
        - name: POSTGRES_USER
          value: "admin"
        - name: POSTGRES_PASSWORD
          value: "admin123"
        - name: POSTGRES_DB
          value: "sampdb"
```

Redis

```
apiVersion: v1
kind: Service
metadata:
  name: redis-service
spec:
  selector:
    app: redis
  ports:
    - protocol: TCP
      port: 6379
```

```
    targetPort: 6379
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: redis
  labels:
    app: redis
spec:
  replicas: 1
  selector:
    matchLabels:
      app: redis
  template:
    metadata:
      labels:
        app: redis
    spec:
      containers:
      - name: redis
        image: redis:7
        ports:
        - containerPort: 6379
```

Nginx

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:latest
        ports:
```

```
        - containerPort: 80
```

Busybox

```
apiVersion: v1
kind: Pod
metadata:
  name: busybox
spec:
  containers:
  - name: busybox
    image: busybox
    command: ["sleep", "3600"]
    imagePullPolicy: IfNotPresent
```

8 Push yaml to Github

Sunday, 7 September 2025 2:11 pm

1. Download Git https://git-scm.com/downloads/win

Open git bash or use ur terminal. To use git via ur terminal, do the following.

Add Git to System PATH

1. **Search for "Environment Variables"** in Windows Start Menu
   Open: "Edit the system environment variables"
2. In the **System Properties** window:
   - Click **"Environment Variables…"**
3. Under **System variables**, find and **select** Path, then click **Edit…**
4. Click **New**, and paste the Git bin path.
   The default is usually one of:
   - C:\Program Files\Git\bin
   - or C:\Program Files\Git\cmd
     If you're unsure, open Git Bash and run:

     where git

     and copy the path.
5. **Click OK** on all dialogs to save.
6. **Restart VS Code completely** (close and reopen).
7. Open the VS Code terminal (PowerShell or CMD) and try:

   git --version

2. Initialise git

```
git init
```

3. Create a .gitignore (optional but good practice)
   Create a .gitignore file if needed (e.g., if you don't want to commit .env files or logs):

   echo "*.env" >> .gitignore

   - This will create a .gitignore file - a file that tells Git which files and folders to skip when committing changes. These files wont be tracked, added or pushed to your repo
   - Examples of some files are r*.log, .env (secrets, PW, API keys), venv

4. Save your github repo URL so you can later push/pull from it easily
   By convention, the first remote connection is called origin. (origin is the nickname for your repo)

- Set remote named origin

```
git remote add origin https://github.com/mikiangelica/k8s-manifests.git
```

- If you want to rename:

```
git remote set-url origin  https://github.com/mikiangelica/k8s-manifests.git
```

- If you want to add a new remote (with new name)

```
git remote add k8s https://github.com/mikiangelica/sampapp.git
```

- Check connections:

```
git remote -v
```

Would show somethjing like the below:
git remote -v
origin https://github.com/mikiangelica/k8s-manifests.git (fetch)
origin https://github.com/mikiangelica/k8s-manifests.git (push)

- This just shows where those actions would go if you did run them

5. Set Github credentials

```
git config --global user.name mikiangelica
git config --global user.email mikiangelica@gmail.com
```

6. Push files to Github

- Be in the correct folder (in the folder with the YAML manifests are)
- Add your YAML files to Git - git add tells Git that you wld like the following files to be tracked and included in the next commit
  - Add all files in current folder

    ```
    git add .
    ```

  - Add specific files

    ```
    git add webapp.yaml redis.yaml
    ```

- Commit your changes - git commit saves a snapshot of all the staged files into Git's local history - but only locally (in ur computer), still doesn't send anyth to github yet; only in ur local repo.

```
git commit -m "Initial commit of k8s YAML manifests
```

- Push to Git

```
git push -u origin master
```

*-u is --set-upstream, so it sets the default remote and branch for your local branch. So it will push your local master branch to the origin remote (connects remote conn to github repo URL) and sets the upstream tracking ref so in future you can just type git push/pull instead of git push/pull origin master

To check which branch youre in rn

```
git branch
```

*What are Git Branches?

- A branch is like a parallel Universe of your project
- It lets you work on features, bug fixes or experiments separately from the main (or master) branch
- Each branch is an independent line of development
- When youre done with the changes in that branch, you can merge it back into main
- Why use multiple Git branches?
  - Feature dev - feature/add-login - work on new features without disturbing main code
  - Bug Fix - fix/postgres-connection - isolates fixes so you can test safely
  - Excperiment - test/redis-cluster - Try new ideas without risking the main app
  - Integration or release - release/v1.2 - prepare prod releases separately

- Hotfix - hotfix/broken-prod - Urgent fixes to prod env

To see default branch for the respective remote name:

```
git remote show origin
```

To switch branch

```
git checkout main
```

*Switches you to the main branch

To check branches

```
git branch     # local branches
git branch -r   # remote branches
git branch -a   # local + remote
```

7. New Branch

- Create and change to the new branch

```
git checkout -b calico-istio-testing
```

How the new branch will look:

```
# git branch    # adds local branch calico-istio-testing
* calico-istio-testing
  master
# git branch -a   # adds local branch calico-istio-testing
* calico-istio-testing
  master
  remotes/origin/master
# git branch -r  * no change as not connected to git yet
  origin/master
```

Netshoot yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: netshoot
  namespace: default
  labels:
    app: netshoot
spec:
  containers:
  - name: netshoot
    image: nicolaka/netshoot:latest
    command: [ "sleep", "3600" ]   # keeps the pod running for testing
  restartPolicy: Always
```

- Add, Commit and push to github

```
git add netshoot.yaml
git commit -m "Add netshoot pod for Calico/Istio testing"
git push -u origin calico-istio-testing
```

*You will see on github UI, that the new branch is created with your files

- Open pull request on github

To check all commits:

```
git log --oneline --graph --all
* 38c4c3e (HEAD -> calico-istio-testing, origin/calico-istio-testing) Add netshoot pod for Calico/Istio testing
* 1fb2bb4 (origin/master, master) Initial commit of k8s YAML manifests
```

Changes to any one file
If you make a change to any file and want to push it to Git

- Edit your file
- Change to respective branch, add, commit, push

```
git checkout master
git add nginx.yaml
git commit -m "Change to nginx.yaml"
git push
```

*The change shld reflect on github itself

Revert changes above

- Commit and push

```
git revert HEAD     # creates a new commit that undoes changes from last one
git push   (if in correct branch)
git push origin master    (if diff branch)
```

8. Merge branch into master
   As of now, we have pushed the new netshoot pod yaml manifest to github under the new branch calico-istio-testing but it is not yet merged into master branch.

- On Github UI, click on "Compare & pull request"
- Review changes and click "Create pull request"
- Click "Merge pull requests"

9. Delete branch after merge:

- On Github (remote)
  - Just click on Delete branch on the branch
  - OR on CLI: # git push origin --delete calico-istio-testing
    *FYI after deleting, if u do git branch -r it will still show the deleted branch bc Git hasn't updated your local list of remote branches yet (it is a cached view)
    To refresh:
    # git fetch --prune
    To make it auto-prune every fetch:
    # git config --global fetch.prune true
- On Local Machine

- # git checkout master # get out of the branch u wna delete
- # git branch -d branch_name (e.g. git branch calico-istio-testing)

9 Access k3s cluster from local

Sunday, 7 September 2025 9:29 pm

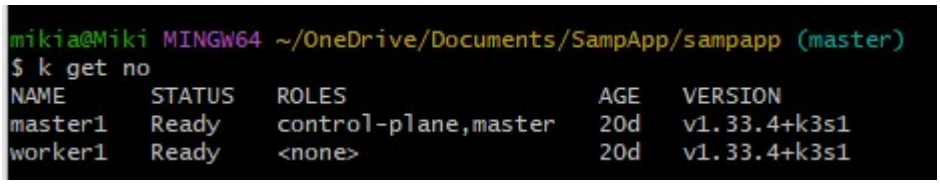
1. Download the latest .exe for Windows:
   - Example: https://dl.k8s.io/release/v1.30.1/bin/windows/amd64/kubectl.exe

2. Move kubectl.exe to a folder (e.g., C:\kubectl) and add that folder to your **System PATH**:
   - Right-click on *This PC* → Properties → Advanced system settings → Environment Variables → Edit Path
3. Confirm installation:

   kubectl version --client

4. Make .kube dir
   1. mkdir %USERPROFILE%\.kube
   2. cd %USERPROFILE%\.kube
   3. SSH into master
      a. Sudo su -
      b. Cp /etc/rancher/k3s/k3s.yaml /home/miki/k3s.yaml
      c. Chmod +r /home/miki/k3s.yaml (actly I did 777)
      d. Chown miki:miki /home/miki/k3s.yaml
   4. Back on local cmd prompt:

      Scp miki@192.168.81.128:/home/miki/k3s.yaml %USERPROFILE%/.kube/config
5. Edit .kube/config file so that the IP is the master node's IP
6. Save and test with kubectl get no
7. **Bonus… I like to do on Git Bash. Easier.



10 Deploy YAMLs on cluster

Sunday, 7 September 2025 10:13 pm

1. Git clone

On the local term (git bash), do:

```
# git clone https://github.com/mikiangelica/k8s-manifests.git
```

Now check there shld be a k8s-manifests folder with all the yaml manifests.

2. Create objects

```
kubectl apply -f .
```

3. Validate

```
kubectl get po -owide
```

11 Calico

Sunday, 7 September 2025 10:23 pm

Technology:
Calico (CNI)

Calico
- East-west traffic (Internal)
- Pod to Pod (L3)
- Without calico, pods can't even talk to each other at all
- Can control which pod talks to who by label/NS/IP (netpols)

Calico GlobalNetpol
- Can controll egress (what traffic leaves cluster e.g. you can control if you don't want pods/svcs to communicate with outside world except GitHub)

Ingress
- North-south traffic (to external)
- Only controls ingress (traffic inbound)
- Cannot control egress

ServiceMesh Istio
- North-south traffic
- Replaces Ingress
- mTLS
- More granular control over traffic and how traffic moves
- Metrics

1. Check for flannel

Journalctl -u k3s | grep flannel # there shld be flannel output if there is embedded flannel
Ls /var/lib/rancher/k3s/agent/etc/cni/net.d/ # shld have a 10-flannel.conflist file

2. Test current connectivity

k exec -it netshoot -- sh
Ping <pod-IP>
Curl <pod-IP>:<port>

*redis:6379, postgres:5432, sampapp:5000, nginx:80

3. Drain both nodes
4. Stop k3s svc

Master node: systemctl stop k3s
Worker node: systemctl stop k3s-agent

5. Disable and remove flannel

Rm -rf /var/lib/rancher/k3s/agent/etc/cni # delete cni directories Rm -rf /etc/cni/net.d # doesn't exist Rm -rf /opt/cni # doesn't exist Rm -rf /var/lib/rancher/k3s/agent/etc/flannel (can optionally delete -- AFTER disabling flannel)
Ip a | grep flannel

Ps aux | grep flannel # check that flannel is no longer running
Ip link delete flannel.1

Edit the /etc/systemd/system/k3s.service.env (master) OR /etc/systemd/system/k3s-agent.service.env (worker) file:
Add in : K3S_NO_FLANNEL=true

6. Start k3s svc

Master node: systemctl start k3s
Worker node: systemctl start k3s-agent

Seems like flannel kept coming back after each restart of k3s. So I decided to just uninstall k3s and reinstall with calico

1. Uninstall k3s and cleanup
   Master Node:

sudo /usr/local/bin/k3s-uninstall.sh

   Worker Node:

sudo /usr/local/bin/k3s-agent-uninstall.sh

   Clean up:

sudo rm -rf /etc/rancher /var/lib/rancher /var/lib/kubelet /etc/cni /opt/cni /var/lib/cni
Rm -rf ~/.kube/config (if u wna reset cluster config on ur local)

2. Install k3s without flannel
   Master node:

curl -sfL https://get.k3s.io | INSTALL_K3S_EXEC="server --flannel-backend=none --disable-network-policy" sh -

   Worker node:

curl -sfL https://get.k3s.io | K3S_URL=https://192.168.81.128:6443 K3S_TOKEN="K1080e8d3f7b27ad61ef4ca4d60c0f53705c52f3aeb3c17a4d782cedcdd865c1430::server:60f6d6681f7e54098d82c7f01d32f64b" sh -

   Confirm that there is no flannel:

Ip a | grep -i flannel
Ps aux | grep flannel

3. Push calico to Git
   In Git bash, go into the SampApp/k8s-manifest and create new branch

Git checkout -b calico

   Next, download calico from browser
   1. Go to:
      https://raw.githubusercontent.com/projectcalico/calico/v3.30.3/manifests/calico.yaml

2. Press Ctrl + S to save the file as calico.yaml
3. Move it to your k8s-manifests Git repo directory

Push to git

Git add calico.yaml
Git commit -m "Add calico yaml"
Git push -u origin calico # create new branch in git and push

Delete calico branch
*Delete on UI

Git fetch --prune (will match UI and remove calico from remote branch)
Git pull origin master (update new merged master branch)
Git branch -d calico # make sure ur not in calico branch

4. Download calico
   I still cant kgn or kgp because k3s cannot fully initialise because there is no CNI.
   So I could either use the calico.yaml from git or download directly to master:

curl -LO https://raw.githubusercontent.com/projectcalico/calico/v3.30.3/manifests/calico.yaml

   Apply calico

KUBECONFIG=/etc/rancher/k3s/k3s.yaml kubectl apply -f calico.yaml --validate=false

   Check on calico pod progress (1 controller and 1 node pod per node)

KUBECONFIG=/etc/rancher/k3s/k3s.yaml kubectl get pods -A

5. Config for master node to access cluster
   Btw add this line to end of .bashrc then source the file.

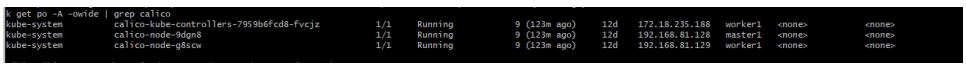
KUBECONFIG=/etc/rancher/k3s/k3s.yaml

   Now test with kubectl get no

6. Config for local to access cluster
   Copy contents of /etc/rancher/k3s/k3s.yaml from master node to your local's ~/.kube/config

Sudo su - and copy contents of above file into /home/miki/k3s.yaml
Change ownership and permissions and scp into local and change IP to master node IP.

   Calico shld be present now.



   *There was a network disruption and nginx, postgres and netshoot pods had ImagePullBackoff error. I reconnected and ran "sudo k3s ctr image pull docker.io/library/nginx:1.25" and now all 3 pods are running. Might be because I reconnected to network

12 Network Policies

Monday, 8 September 2025 8:56 pm

Default Deny-all Ingress

```
# netpol-default-deny.yaml

apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: default-deny-all
  namespace: default
spec:
  podSelector: {}
    matchLabels:
      role: db
  policyTypes:
  - Ingress
```

Netshoot to Sampapp

```
# netpol-allow-netshoot-to-sampapp.yaml

apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-netshoot-to-sampapp
  namespace: default
spec:
  podSelector:
    matchLabels:
      app: sampapp
  policyTypes:
  - Ingress
  ingress:
  - from:
    - podSelector:
        matchLabels:
          app: netshoot
```

Sampapp to db

```
# netpol-allow-sampapp-to-db.yaml


apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-sampapp-to-db
  namespace: default
spec:
  podSelector:
```

```
    matchLabels:
      app: redis
  policyTypes:
  - Ingress
  ingress:
  - from:
    - podSelector:
        matchLabels:
          app: sampapp
---
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-sampapp-to-postgres
  namespace: default
spec:
  podSelector:
    matchLabels:
      app: postgres
  policyTypes:
  - Ingress
  ingress:
  - from:
    - podSelector:
        matchLabels:
          app: sampapp
```

Busybox to nginx

```
# netpol-allow-busybox-to-nginx.yaml

apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-busybox-to-nginx
  namespace: default
spec:
  podSelector:
    matchLabels:
      app: nginx
  policyTypes:
  - Ingress
  ingress:
  - from:
    - podSelector:
        matchLabels:
          app: busybox
```

Policies made:

1. Default Deny-all Ingress
   Go into the netshoot pod

   ```
   Kubectl exec -it netshoot -n default -- sh
   ```

   Try to connect to redis and postgres (shldnt work)

   ```
   Nc -zv redis-service 6379
   ```

```
Nc -zv postgres 5432
```

2. Netshoot to Sampapp
   Go into the netshoot pod

```
Kubectl exec -it netshoot -n default -- sh
```

   Try to connect to sampapp (*shld work)

```
Curl sampapp-service:5000
```

3. Sampapp to db (redis and postgres)
   **Btw I edited the sampapp yaml here so that there is a netshoot container in it.
   ***Make sure to specify container below or it will default to first container sampapp

```
Kubectl exec -it sampapp-pod -n default -c netshoot -- sh
Nc -zv 172.16.235.136 6379  (redis)
Nc -zv 172.16.137.73 5432  (postgres)
```

   Ran into an issue where it could not reach redis and postgres. The labels and pods were all correct. Turns out dns issue because it could reach the pod Ips:

```
~ # nc -zv 172.16.235.136 6379
Connection to 172.16.235.136 6379 port [tcp/redis] succeeded!
~ # nc -zv 172.16.137.73 5432
Connection to 172.16.137.73 5432 port [tcp/postgresql] succeeded!
```

   **After some troubleshooting I realise it is not a dns issue as Im able to use nslookup to the services. It is not a netpol issue perse because I can connect via the pod IPs - but because I try to connect via the service (which means going through the service cluster IP that forwards traffic to the relevant pod) -- so it is not directly trying to connect to the pod with the specified label in my netpol.

4. Busybox to nginx
   Make sure there is a nginx-service there and exec into busybox pod

```
Kubectl exec -it busybox -- sh
```

   Connect to nginx

```
Wget --spider nginx-service
```

   OR

```
Wget nginx-service
```

   Expected output:

```
/ # wget --spider nginx-service
Connecting to nginx-service (10.43.104.188:80)
remote file exists
/ # wget nginx-service
Connecting to nginx-service (10.43.104.188:80)
saving to 'index.html'
index.html         100%
|*******************************************************************************
*************************************************************|   615  0:00:00
ETA
```

```
'index.html' saved
```

13 PVC for postgres

Saturday, 13 September 2025 11:50 pm

Technology:
Postgres in K3s

We use PVCs for services that are stateful, like Postgres.
Stateful = Keeps important data that must persist across restarts

E.g.

Postgres > Stores databases/tables
MySQL > Stores user records
Kafka/Zookeeper > Store offsets/metadata

Stateless = Can be killed/restarted anytime without losing critical data (e.g. Nginx, frontend API)

They require persistent storage so that they have a stable, durable volume that survives pod restarts and migrations… OTHERWISE the data is stored on the pod's ephemeral (short term) container filesystem and if the pod restarts, all data is LOST!!
A.K.A Without PVC, your Postgres DB would vanish every time the pod dies.

To bind a PVC to postgres, we need a storageclass (SC), Persistent Volume (PV), and Persistent Volume Claim (PVC).

How they work together:

1. You define storage class (e.g. gp2)
2. Postgres PVC says "I need 10Gi from gp2)
3. K8s (via CSI driver) provisions a PV (e.g. EBS Volume)
4. The PVC is bound to that PV

Creating the PVC for Postgres

1. First, we check if there is an SC in the cluster with:

```
# kubectl get sc


NAME                 PROVISIONER             RECLAIMPOLICY   VOLUMEBINDINGMODE
ALLOWVOLUMEEXPANSION   AGE
local-path (default)   rancher.io/local-path   Delete          WaitForFirstConsumer   false                  6d6h
```

This SC will dynamically create PVs for you whenever you make a PVC so don't need create PV.
Most times don't need to create PV unless in static provisioning cases… like u want to handcraft the PV and manually bind to a PVC etc.

-- Rule of thumb > If an SC exists in cluster, only need to create PVC. If no SC, need to create SC, PV and PVC

3. Replace postgres yaml from a deployment to a statefulset
   Statefulsets:
   - Guarantee stable pod names e.g. postgres-1 postgres-2
   - Creates PVCs per replica using VolumeClaimTemplates
   - Ensures the pod reattaches to the same volume when rescheduled.

```yaml
apiVersion: v1
kind: Service
metadata:
  name: postgres-service
  namespace: default
spec:
  selector:
    app: postgres
  ports:
  - port: 5432
---
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: postgres
  namespace: default
spec:
  selector:
    matchLabels:
      app: postgres
  serviceName: "postgres"
  replicas: 1
  template:
    metadata:
      labels:
        app: postgres
    spec:
      containers:
      - name: postgres
        image: postgres:15
        env:
        - name: POSTGRES_PASSWORD
          valueFrom:
            secretKeyRef:
              name: postgres-secret
              key: password
        ports:
        - containerPort: 5432
        volumeMounts:
        - name: postgres-storage
          mountPath: /var/lib/postgresql/data
  volumeClaimTemplates:
  - metadata:
      name: postgres-storage
    spec:
      accessModes: [ "ReadWriteOnce" ]
      storageClassName: "local-path"
      resources:
        requests:
          storage: 1Gi
```

Push above yaml manifest to github… create new branch, add, commit, push. Then create pull req on github and merge branch. Then back on CLI, git fetch --prune and git branch -d postgres.

Apply the new manifest…

Faced multiple issues from here:

ISSUE 1: Pod, PVC and PV not coming up
CAUSE: Provisioner pod wasn’t even running - causing PVCs to be stuck in Pending state and failing to provision
FIX: Redeployed the local-path-provisioner into cluster by reapplying manifest
Troubleshooting Steps Overview: Noticed Pod was not coming up. Checked PVC and PV and realised they weren't coming up either. Then checked the provisioner pod (responsible for creating the local PVs based on PVC requests) and saw it wasn't running.
Reference: Troubleshooting Log 2

ISSUE 2: Pod still not coming up with 'calico' failure error (ContainerCreating state) - Calico Pods Issue
CAUSE: Calico was running on BGP mode.
FIX: Switched to VXLAN mode and reapplied VXLAN calico manifest with same initial version and these network interface calico errors cleared
Troubleshooting Steps Overview: Checked logs of pod, and saw that it had network interface (calico) errors, Checked calico pods & their logs, cluster status. Tested connectivity with netshoot and got same calico error. Reapplied calico manifest but same error observed. Checked if CNI binaries exist on worker node (calico, calico-ipam) for kubelet to call calico. Checked connectivity to API server (connected). Checked for CNI config file and calico crd existence (both fine). Checked kubeconfig file in worker node and did a curl to the ClusterIP API (reachable). Reapplied calico manifest again and now calico pods are not running. Checked logs and noticed it says BGP is not ready. Switched to VXLAN mode by editing default IPPool, reapplied calico manifest with vxlan mode with correct calico version and restarted pods.
Reference: Troubleshooting Log 2

ISSUE 3: CoreDNS failure found
CAUSE: Config map misconfiguration
FIX: Amended Config map
Troubleshooting Steps Overview: During pod to pod connectivity test in nettest ns, ping was failing by pod name - but via pod IP was fine. Checked coreDNS pod and svc. Checked DNS from test pods and logs of coredns pod, which showed that coreDNS is forwarding queries upstream to possibly host VM's DNS which is not reachable from inside CoreDNS pod. Amended coredns configmap so that it forwards to 8.8.8.8 1.1.1.1 instead. Restarted deployment and DNS was working.
Reference: Troubleshooting Log 2

ISSUE 4: (RELATED TO ISSUE 1) Local-path-storage pods not coming up
CAUSE: Broken config.
FIX: Reapply manifests and also create secret.
Troubleshooting Steps Overview: All local-path-storage pods were CrashLoopBackOff. Checked logs and saw error: cm is missing in ns. Removed finalizers block from ns config and force deleted broken namespace. Had to apply ns.json via API finalize endpoint to rly delete ns as it was stuck. Reapplied local-path-storage manifest and all local-path-storage pods are up - as well as the sc, pvc, pv of postgres. But postgres pod still CreateContainerConfigError. Checked logs and saw that it could not find postgres-secret secret. Had to create secret and verify. Restarted pod and all is well.
Reference: Troubleshooting Log 2

14 Cronjob for backups for Postgres

Tuesday, 16 September 2025 12:09 am

Over here we create a cronjob that connects to Postgres DB using the Secret created for it (matches postgres credentials), and collects dumps to store in the pvc mount /backups. This /backups is located in 1 of the worker nodes in my lab cluster - but in Prod clusters, this /backups is located in a dedicated DB node for the cluster. Hence why we have DB nodes.

1. Create pvc

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pg-backup-pvc
  namespace: default
spec:
  storageClassName: local-path
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
~
```

```
# k apply -f pg-backup-pvc.yaml
```

2. Create secret

```
 kubectl create secret generic pg-backup-secret \
> --from-literal=POSTGRES_USER=postgres \
> --from-literal=POSTGRES_PASSWORD=admin123 \
> --from-literal=POSTGRES_DB=postgres
secret/pg-backup-secret created
```

*The POSTGRES_USER, PASSWORD, DB here must match the ones Postgres is actually running with (from STS secret) or else pg_dump will fail to connect.
This secret (pg-backup-secret) is for clients (like your backup job) to connect to the running Postgres DB.

3. Create cronjob

```
apiVersion: batch/v1
kind: CronJob
metadata:
  name: pg-backup
  namespace: default
spec:
  schedule: "0 2 * * *"    # Every day at 2 AM
  jobTemplate:
    spec:
      template:
        spec:
          containers:
```

```
      - name: pg-backup
        image: postgres:15
        imagePullPolicy: IfNotPresent
        envFrom:
          - secretRef:
              name: pg-backup-secret
        volumeMounts:
          - name: backup-storage
            mountPath: /backups
        command:
        - /bin/sh
        - -c
        - |
          FILE=/backups/backup-$(date +\%F-\%H%M).sql
          PGPASSWORD=$POSTGRES_PASSWORD pg_dump -h postgres-service -U $POSTGRES_USER
$POSTGRES_DB > $FILE
          echo "$Backup saved to $FILE"
      restartPolicy: OnFailure
      volumes:
        - name: backup-storage
          persistentVolumeClaim:
            claimName: pg-backup-pvc
~
~
```

**Over here, we mount the PVC pg-backup-pvc at /backups.

- The pg_dump command writes SQL dumps into /backups/backup-<date>.sql
- That means the dump file is persisted on the PV provisioned by your local-path SC
- So even if the pod dies, the file stays on disk (wherever your local-path-provisioner stores PVs on the node.. Usually /var/lib/rancher/k3s/storage/…)

After creating the above, PVC was in Pending state as it the SC is WaitForFirstConsumer.
So we created a manual trigger to test the cronjob immediately instead of waiting til 2AM.

- It clones my cronjob into a one time job so I can validate:
  - That the pvc binds successfully
  - the pod runs pg_dump
  - that a backup file gets written into your /backups mount

```
# kubectl create job --from=cronjob/pg-backup pg-backup-test
job.batch/pg-backup-test created
```

After this, I see that pvc is bound and pv is there

```
# k get pvc
NAME                  STATUS   VOLUME                                     CAPACITY   ACCESS MODES
STORAGECLASS   VOLUMEATTRIBUTESCLASS   AGE
pg-backup-pvc         Bound    pvc-87ab800a-aab9-48a8-a2ea-3d57ab5bd9cb   1Gi        RWO
local-path     <unset>                 5m30s
postgres-storage-postgres-0   Bound    pvc-bab4c32a-5989-4662-ac1a-5f32c2f6495d   1Gi        RWO
local-path     <unset>                 22h

# k get pv
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM
STORAGECLASS   VOLUMEATTRIBUTESCLASS   REASON   AGE
pvc-87ab800a-aab9-48a8-a2ea-3d57ab5bd9cb   1Gi        RWO            Delete           Bound    default/pg-
backup-pvc     local-path     <unset>                          2m9s
```

```
pvc-bab4c32a-5989-4662-ac1a-5f32c2f6495d   1Gi        RWO            Delete           Bound
default/postgres-storage-postgres-0   local-path     <unset>                          15h
```

15 GitHub Resync

Thursday, 18 September 2025 5:36 am

I made some changes directly to the cluster a few times during troubleshooting, so for this section I will be rearranging my github repo and taking my latest cluster yamls and configs and pushing them to GitHub.

Restructuring my GitHub repo

Intended Structure

```
SampApp/
├── sampapp/                # All infra + manifests + automation
│   ├── README.md
│   ├── app/                # Your actual app code (Python etc)
│   │   ├── requirements.txt
│   │   ├── app.py
│   │   └── ...
│   ├── manifests/
│   ├── terraform/
│   ├── ansible/
│   ├── Dockerfiles/
│   ├── images/
└── .gitignore
```

Edited Dockerfile

```
FROM python:3.10-slim

ENV PYTHONDONTWRITEBYTECODE=1
ENV PYTHONUNBUFFERED=1

WORKDIR /app

COPY app/requirements.txt .

RUN pip install --no-cache-dir -r requirements.txt

COPY app/ .

EXPOSE 5000

CMD ["python", "app.py"]
```

- Paths are relative to the build context!!
- I run docker build in SampApp/ dir, and when I run docker build I ran with sampapp as my build context (see the end), so in my Dockerfile, when I specify "app/" it is relative to sampapp/

Ran into the following issue:

ISSUE 1: ERROR BUILDING IMAGE
CAUSE: Build context was too big so it cancelled

FIX: To only inclujde sampapp dir (context build)
Troubleshooting Steps Overview: Analysed error message from build command, reduced build context to just application directory
Reference: Troubleshooting Log 1

Resyncing my GitHub repo

kubectl get all -n default -oyaml > manifests/base/default-all.yaml
kubectl get sc,pv,pvc -oyaml > manifests/storage/storage.yaml
kubectl get netpol -Aoyaml > manifests/netpols/netpols.yaml
kubectl get cm -Aoyaml > manifests/base/configmaps.yaml
kubectl get secrets -Aoyaml > manifests/base/secrets.yaml

README
Notes on Extracted Manifests
These manifests are direct exports from my cluster and may include system fields. In a production GitOps workflow, I would clean them using kubectl-net or yq.
Notes on Secrets
Secrets are also exported in raw base64-encoded form for completeness.
In a real production environment, **storing raw secrets in Git is a no-no**
Instead, a team would use Sealed Secrets/External Secrets Operator/HashiCorp Vault/SOPS+GitOps

This repo demonstrates my learning and workflow process. In a prod scenario I would use a proper secret management solution.

Push to Git
git add manifests/ README.md
git commit -m "Add synced cluster YAMLs and updated README notes on secrets"
git push (didn't work because I completely restructured my local dirs)
git push origin master --force (added the new files forcefully)

On Github it still showed the old files…

git ls-files (still showed the old files)
(Checked if files are still actually there - they weren't)
git add .
git rm -r --cached . (untracks everything Git still thinks exists (incl old files)
git add . k(stages all actual files/folders that still exists locally)
git commit -m "Remove old tracked files and sync repo structure"
git push origin master (syncs with github)

Now updated.

16 Ingress

Saturday, 13 September 2025 11:54 pm

Technology:
Ingress

Components in Ingress setup

1. Ingress Controller (ingress-nginx-controller Pod)
    - Traffic Cop / Reverse proxy (NGINX running inside K8s)
    - Watches the Ingress resources in the cluster
    - When the pod starts, it runs with an argument like:

```
--ingress-class=nginx
--controller-class=k8s.io/ingress-nginx
```

    - So the above means that it watches for ingress objects with ingressClassName= nginx and it syncs those rules into its own internal NGINX config
    - Actually accepts requests (on NodePort/LoadBalancer ports like :30644) and decides where to send them
    - It does not talk directly to Pods, it only talks to services
    - Role: 'entry point into your cluster'
2. Ingress Resource
    - Just configuration (rules)
    - Ingress resource will have a field (ingressClassName: nginx), which tells K8s that this ingress belongs to the controller that manages the class nginx.
    - Says things like:
        - If Host: sampapp.local and path / -> Send them to sampapp-service:5000
        - This is how the controller know which Service to target
    - Role: 'traffic rules' for the controller to follow
3. IngressClass (nginx)
    - This is a cluster-wide object that maps a name (nginx) to a specific controller

```
apiVersion: networking.k8s.io/v1
kind: IngressClass
metadata:
  name: nginx
spec:
  controller: k8s.io/ingress-nginx
```

    - It says "Any Ingress with ingressClassName: nginx shld be handled by the Ingress-NGINX controller
4. Service (sampapp-service)
    - Stable front door for your app's Pods
    - Has a ClusterIP (10.x.x.x) and points to Pod endpoints (172.x.x.x)
    - Service has selectors (labels like app=sampapp) which map it to Pods
    - Auto maintains an endpoints object listing all matching pods
    - Role 'load balancer inside the cluster'
5. App Pods (sampapp-xxx_
    - These are the actual Flask containers running your code
    - They bind to port 5000 and serve requests

- Role 'where the real app lives'

Ingress Controller takes instruction from Ingress resource and carries it out. For IngressController and Ingress Resource to connect, they need IngressClass as the medium.

End to end Connection

Client (curl -H "Host: sampapp.local" http://192.168.81.128:30644)
↓
Ingress-NGINX Controller Pod (172.18.137.121:80)
(uses Ingress rules to find the right Service)
↓
Service: sampapp-service (ClusterIP 10.43.133.187:5000)
(maps to Endpoints)
↓
Endpoints → Pod IPs (172.18.137.65:5000, 172.18.137.70:5000)
↓
sampapp Pods (Flask app responds on port 5000)

Flow of a request

Browser (Host: sampapp.local)
↓
NodePort/LoadBalancer (30644) → ingress-nginx-controller Pod
↓
Ingress Resource (rules) says: route Host=sampapp.local to sampapp-service:5000
↓
sampapp-service → forwards to endpoints
↓
sampapp Pods (Flask app on port 5000)

Implementation

1. Download version 3.19.0 of Helm
   https://github.com/helm/helm/releases
2. Moved Helm to an existing path in env variables PATH
3. Install ingress-nginx (controller)
   # helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
   # helm repo update
   # helm install ingress-nginx ingress-nginx/ingress-nginx

Ran into the various issues:

ISSUE 1: Installation failed
CAUSE: Existing resources preventing installation due to config mismatch
FIX: Deleted all current ingress resources
Troubleshooting Steps Overview: Analysed error message from install command, deleted existing ingress resourced and verified helm release is deployed, and ingress pods, svc and resources are up. Mapped host locally (sampapp.local) since no external IP. Tested connectivity and it reached ingress-nginx controller! But next issue…
Reference: Troubleshooting Log 3

ISSUE 2 : Error 503 (Either ingress or svc issue)
CAUSE: Request is reaching ingress-nginx controller but can't find a backend svc to route to (svc/Ingress issue)
FIX: Edited ingress to match port of ingress svc
Troubleshooting Steps Overview: Check ingress and svc port and IP, found a port mismatch [5000 vs 80], Tested connectivity and now no more 503 error… now it is 504 error :(
Reference: Troubleshooting Log 3

ISSUE 3 : 504 error gateway timeout
CAUSE: No response was received within timeout duration and connection closed.
FIX: Deleted all evicted pods
Troubleshooting Steps Overview: Checked svc endpoints, pod IPs and labels. Svc and Endpoints were fine but there were many failed pods. Deleted all evicted pods til left with 2 running pods - matched EP IPs to these 2 running pods. Tested curl again and still 504 error.
Reference: Troubleshooting Log 3

ISSUE 4 Ingress svc external IP pending, svc loadbalancer helpers failing, local-path-provisioner is broken
CAUSE: No external IP for Ingress service, causing loadbalancer helpers to fail and local-path-provisioner to break
FIX: Quick workaround - use NodePort
Troubleshooting Steps Overview: Edited ingress-nginx-controller svc to change type to NodePort. Tested connectivity from inside ingress-controller pod and still got 504 error.
Reference: Troubleshooting Log 3

ISSUE 5 Nodes unhealthy (CPU and disk)
CAUSE: Pressure from disk also causing CPU pressure
FIX: Cleared disk space, then deleted all Failed pods (ephemeral storage) which alleviated CPU pressure too
Troubleshooting Steps Overview: Checked node health and saw that they had disk and CPU pressure. Cleared disk space, deleted failed pods and verified healthy Node pressure again
Reference: Troubleshooting Log 3

ISSUE 5 FIXED

Now back to checking ingress

1. Check node health

```
kubectl get nodes -o wide    (all shld be Ready)
kubectl describe node master1 | grep -A5 Conditions   (all shld be False)
kubectl describe node worker1 | grep -A5 Conditions   (all shld be False)
```

2. Check that all ingress controller pods are running

```
# kubectl get pods -n ingress-nginx -o wide
NAME                                   READY   STATUS    RESTARTS   AGE   IP
NODE     NOMINATED NODE   READINESS GATES
ingress-nginx-controller-6b96cf684f-qz8s9   1/1     Running   0          20m
172.18.137.121   master1   <none>           <none>
```

3. Describe pod

```
# kubectl describe pod -n ingress-nginx <ingress-nginx-pod-name>
```

4. Check Ingress object

```
# kubectl get ingress -n default
# kubectl describe ingress sampapp-ingress -n default
```

5. Check that service ports for ingress are 80:xxxxxx/TCP and 443:yyyyyy/TCP

```
# k get svc -n ingress-nginx
NAME                                 TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)
AGE
ingress-nginx-controller             NodePort    10.43.230.85    <none>
80:30644/TCP,443:30589/TCP   10h
ingress-nginx-controller-admission   ClusterIP   10.43.251.154   <none>
443/TCP                      10h
```

Take note of the NodePort

```
# kubectl get svc sampapp-service -n default -oyaml | egrep "port:
|targetPort:"
# kubectl get ep sampapp-service-n defualt -owide
```

Shld be service port 5000/TCP, ep shld be IPs of sampapp pods

6. Check backend pods

```
# kubectl get pods -n default -l app=sampapp -o wide
# kubectl logs -n default <one-sampapp-pod> -c sampapp
```

7. Test Ingress from inside cluster

```
# kubectl run curltest --rm -it --image=curlimages/curl --restart=Never -- \
  curl http://sampapp-service.default.svc.cluster.local:5000
```

And from outside

```
# curl -H "Host: sampapp.local" http://<node-ip>:<nodeport>
```

Curl was still not working!

ISSUE 6 Netpol
CAUSE: My test netpols were blocking traffic!
FIX: Deleted all netpols (will be implementing properly in future)
Troubleshooting Steps Overview: Was checking connectivity from ingress-nginx pod when I remembered the netpols, then deleted all netpols. Connectivity is now working.
Reference: Troubleshooting Log 3

Push ingress yamls to Github

```
# kubectl get ingress -A -o yaml > ingress/all-ingresses.yaml
# kubectl get ingressclass -o yaml > ingress/all-ingressclasses.yaml
# kubectl get ingress sampapp-ingress -n default -o yaml >
ingress/sampapp-ingress.yaml
```

Went back to sampapp dir and:

```
# git add .
# git commit -m "Update with Ingress"
# git push
```

17 CI/CD w Jenkins & ArgoCD

Saturday, 13 September 2025 11:54 pm

Technology:
GitOps
GitHub - Source of Truth for manifests
CI - Jenkins (build image)
CD - ArgoCD (applies manifests)

Pros: Auditable, GitOps best practice, clean separation of roles
Cons: More moving parts (Jenkins + ArgoCD)

Jenkins is a CI/CD Tool. It can build and deploy, which makes things simple if we just have Jenkins. It will:

- Checkout repo > Build code
- Build & Push docker image
- Update your deployment YAML with new image tag
    - e.g. patch sampapp-deployment.yaml image to myrepo/sampapp:<newtag>
- Commit and Push that YAML change back to GitHub
- Apply k8s manifests directly to cluster (Jenkins needs a kubeconfig with access to cluster)

But generally this is not ideal for GitOps. Jenkins has cluster access. It is harder to audit and rollback as manifests may drift from GitHub.

With a CD Tool like ArgoCD, Jenkins will only have the CI role:
Jenkins will:

- Checkout repo > Build code
- Build & Push docker image
- Update your deployment YAML with new image tag
    - e.g. patch sampapp-deployment.yaml image to myrepo/sampapp:<newtag>
- Commit and Push that YAML change back to GitHub

ArgoCD will:

- Detect the change in GitHub
- Sync the new manifest into the cluster

So with this setup:

- Jenkins (CI+CD) -> Jenkins directly t

Jenkins (CI side: build & push image)

1. Using Helm, deploy Jenkins on cluster

```
# helm repo add jenkins https://charts.jenkins.io
# helm repo update
# helm install jenkins jenkins/jenkins -n jenkins --create-namespace
```

2. Got Jenkins admin pw (on windows via Git Bash)

```
# kubectl get secret jenkins -n jenkins -o jsonpath="{.data.jenkins-admin-password}" | base64 --decode
```

```
&& echo
```

3. Expose Jenkins via Port-forward (easiest and fastest for dev/test)
   Later once ingress-nginx is stable, can expose Jenkins behind an Ingress rule (jenkins.local).

```
# kubectl port-forward svc/jenkins -n jenkins 8080:8080
```

4. Login to Jenkins :
   Open http://localhost:8080 in browser.
   Username: admin
   PW: the PW above

5. Install recommended Plugins in Jenkins
   Must-Have Plugins for Your Setup
   1. Core CI/CD
   - **Pipeline** ☑ → lets you use Jenkinsfile for declarative pipelines.
   - **Git** → integrates with Git repositories.
   - **GitHub Integration** → enables GitHub webhooks, builds on push.
   - **GitHub Branch Source** → auto-detects branches and PRs.

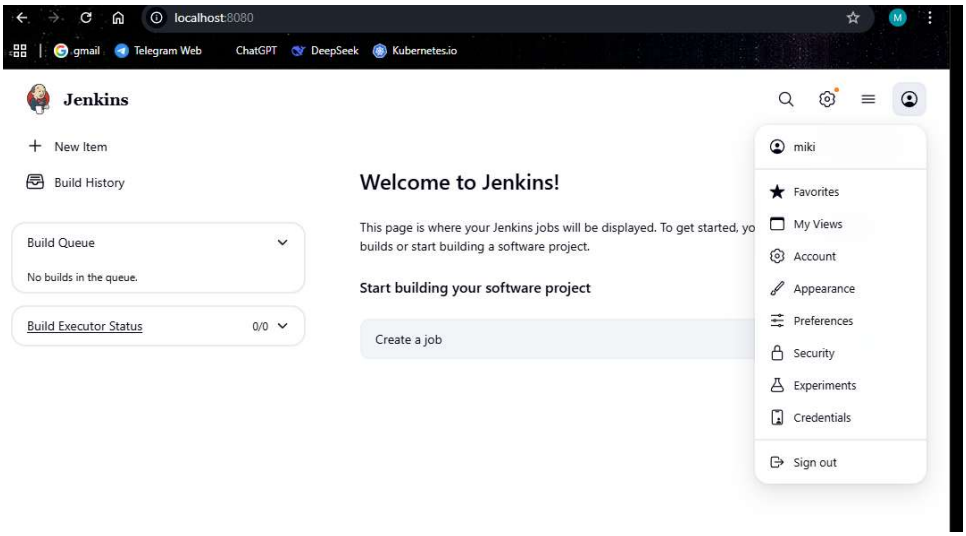
   2. Container & Kubernetes
   - **Docker Pipeline** → allows docker.build() and docker.push() inside Jenkinsfile.
   - **Kubernetes** (a.k.a. Kubernetes plugin) → run Jenkins agents as pods in your cluster (optional, but very useful).
   - **Kubernetes Credentials Provider** → stores secrets in Kubernetes and syncs them into Jenkins.

   3. Credentials & Secrets
   - **Credentials Binding** → lets you use Docker registry passwords, GitHub tokens, etc. safely in pipelines.
   - **Config File Provider** → store kubeconfigs, YAMLs, or .npmrc securely for pipelines.

   4. Notifications / Quality of Life
   - **Blue Ocean** → a modern UI for pipelines (optional, but makes pipelines much clearer).
   - **Email Extension** → send pipeline result emails.
   - **Slack Notification** (if you use Slack).

   How the plugins Fit Into Your Flow
   a. **GitHub Plugins** → Jenkins watches your sampapp repo.
   b. **Pipeline + Docker Plugins** → Jenkins builds/pushes Docker image to your registry.
   c. **Kubernetes Plugin** → optionally, Jenkins jobs run in ephemeral pods inside your k8s cluster.
   d. **Credentials Binding** → safely pass registry login (e.g., DockerHub password, GitHub PAT).
   e. **Config File Provider** → handle kubeconfigs or ArgoCD CLI tokens.
   f. **ArgoCD** → not a Jenkins plugin; ArgoCD just watches your repo for updated manifests. Jenkins doesn't deploy — it only commits manifest changes.

6. Created new admin user
   Manage Jenkins > Users > Create User



ArgoCD (CD - Deploy)

1. Install ArgoCD

```
# kubectl create namespace argocd
# kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml
# kubectl -n argocd get pods
# all should become Running/Ready (api-server, repo-server, server, dex, redis, notifications)
```

2. Access ArgoCD UI

```
# kubectl -n argocd port-forward svc/argocd-server 8080:80
```

Go to http://localhost:8080

3. Get the initial admin password

```
# kubectl -n argocd get secret argocd-initial-admin-secret \
  -o jsonpath="{.data.password}" | base64 --decode && echo
```

ISSUE 1: EPHEMERAL STORAGE OF NODES CAUSING ALL PODS TO FAIL
CAUSE: Ephemeral storage was very high, causing CPU pressure too.
FIX: Capacity Add (Disk)
Troubleshooting Steps Overview: All pods were failing with error messages of ephemeral storage issues. Checked node health and this confirmed it. Evaluated and confirmed that disk space needed to expand to accommodate more workloads. Expanded disk and restored cluster to working state.
Reference: Troubleshooting Log 4

ISSUE 2: LOCAL-PATH-PROVISIONER-PODS IN CRASHLOOPBACKOFF (RBAC)
CAUSE: Local-path-provisioner does not have RBAC permission to read the local-path-config CM in its namespace
FIX: Gave service account the correct RBAC role
Troubleshooting Steps Overview: While checking status of all pods, saw this pod failing and checked logs. Verified that all PVs, PVCs are fine. Gave SA the correct RBAC role and restarted pods.
Reference: Troubleshooting Log 4

Continue with installing ArgoCD

1. Delete argocd namespace with all its resources (since there are many broken argoCD pods due to the ephemeral storage issue last time)

```
# k delete ns argocd --grace-period=0 --force
```

2. Create ns again

```
# k create ns argocd
```

3. Apply official argocd manifest

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

This installs all core components:

- argocd-server (UI + API)
- argocd-repo-server (git ops)
- argocd-application-controller (reconciler)
- argocd-dex-server (SSO/auth)
- argocd-redis (caching)

4. Verify pods are running

```
# kubectl get pods -n argocd
NAME                                  READY   STATUS    RESTARTS   AGE
argocd-application-controller-0       1/1     Running   0          57s
argocd-applicationset-controller-54f96997f8-m2bjb   1/1     Running   0          57s
argocd-dex-server-798cbff4c7-pg447    1/1     Running   0          57s
argocd-notifications-controller-644f66f7df-qw6tc    1/1     Running   0          57s
argocd-redis-6684c6947f-2lkwq         1/1     Running   0          57s
argocd-repo-server-6fccc5759b-lz7hl   1/1     Running   0          57s
argocd-server-64d5fcbd58-8qbdt        1/1     Running   0          57s
```

5. Get ArgoCD initial admin PW

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

6. Since there's no LoadBalancer here, we expose via NodePort:

```
# kubectl patch svc argocd-server -n argocd -p '{"spec": {"type": "NodePort"}}'
service/argocd-server patched

# k get svc argocd-server -n argocd
NAME          TYPE       CLUSTER-IP     EXTERNAL-IP   PORT(S)                      AGE
argocd-server   NodePort   10.43.157.58   <none>        80:31805/TCP,443:32619/TCP   6m8s
```

7. Access ArgoCD UI
   https://<VM-IP>:<Node-Port>
   https://192.168.81.128:31805 (master IP)

Back up entire cluster to GitHub again (before we test out the CI/CD)

1. Application Manifests (GitOps style)

➢ The following will capture all that should run but not the data (PVC contents)

```
# All resources in all namespaces (except cluster-wide CRDs etc.)
kubectl get all --all-namespaces -o yaml > cluster-all.yaml

# ConfigMaps and Secrets
kubectl get cm,secrets --all-namespaces -o yaml > cluster-configs.yaml

# Storage resources
kubectl get sc,pv,pvc --all-namespaces -o yaml > cluster-storage.yaml

# Network policies
kubectl get netpol --all-namespaces -o yaml > cluster-netpols.yaml
```

Set up a GitHub > Jenkins > ArgoCD pipeline

1. **Created a Jenkins file in repo root**

```
pipeline {
  agent any
  stages {
    stage('Checkout') {
      steps {
        git branch: 'master',
          url: 'https://github.com/mikiangelica/k8s-manifests.git'
      }
    }
    stage('Lint Manifests') {
      steps {
        sh 'kubectl kustomize k8s-manifests/ > /dev/null || true'
      }
    }
    stage('Build & Push Image') {
      steps {
        sh '''
        docker build -t mikiangelica/sampapp:${BUILD_NUMBER} .
        docker push mikiangelica/sampapp:${BUILD_NUMBER}
        '''
      }
    }
    stage('Update Manifests') {
      steps {
        sh '''
        sed -i "s|image: .*$|image: mikiangelica/sampapp:${BUILD_NUMBER}|"
sampapp/manifests/app/deployment.yaml
        git config --global user.email "mikiangelica@gmail.com"
        git config --global user.name "jenkins"
        git commit -am "Update image to build ${BUILD_NUMBER}"
        git push origin master
        '''
      }
    }
  }
}
```

The above will create a new pipeline job in Jenkins and connect it to my GitHub rerpo (manifests + app code)

2. **ArgoCD : set up an application**
   Point ArgoCD to the GitHub repo and path with manifests

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: sampapp
  namespace: argocd
spec:
  project: default
  source:
    repoURL: https://github.com/mikiangelica/k8s-manifests.git
    targetRevision: master
    path: manifests
  destination:
    server: https://kubernetes.default.svc
    namespace: default
  syncPolicy:
    automated:
      prune: true
      selfHeal: true
```

   Apply

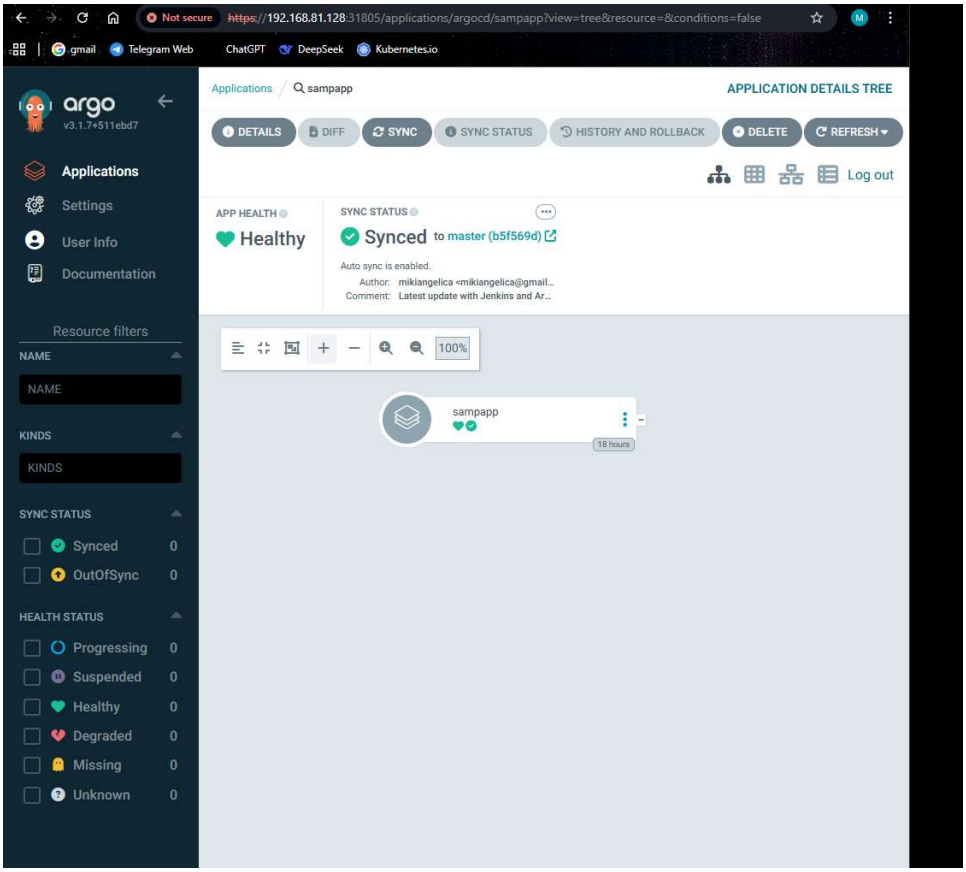
```
# k apply -f argocd-app.yaml
application.argoproj.io/sampapp created

# k get app -A
NAMESPACE   NAME     SYNC STATUS   HEALTH STATUS
argocd      sampapp  Unknown       Healthy
```

   What the columns mean
   - SYNC STATUS
     - Synced → manifests in cluster match what's in GitHub.
     - OutOfSync → something in GitHub differs from cluster.
     - Unknown → ArgoCD hasn't synced yet or doesn't know the state.
   - HEALTH STATUS
     - Healthy → the deployed pods/services are running fine.
     - Progressing → deployment is rolling out.
     - Degraded → something failed (crashloop, not ready).
     - Unknown → ArgoCD hasn't checked health yet.

ISSUE 3 : UNKNOWN SYNC STATUS
CAUSE: argocd Netpol that came with manifest was blocking communication to argocd redis pod
FIX: Deleted all argocd netpols too.
Troubleshooting Steps Overview: Tried various DNS and connectivity tests within argocd NS between argocd repo server pod, argocd application controller and redis. Checked logs and configs of services and pods. Tried to create new netpol to explicitly allow connection but it didn't work. Deleted all netpols
Reference: Troubleshooting Log 4

NOW my cluster sync status is Synced.

```
# k get app sampapp -n argocd
NAME      SYNC STATUS   HEALTH STATUS
sampapp   Synced        Healthy
```



NOW LETS MAKE A PIPELINE.

Push code to GitHub > Jenkins pulls repo, builds Docker image, pushes to DockerHub, commits updated manifests back to GitHub > ArgoCD detects repo changes, syncs manifests and deploys new image to cluster

1. First add credentials to Jenkins for GitHub and DockerHub. We will create the Access Tokens in both before adding to Jenkins.

   In Github UI:
   - Go to **GitHub → Settings** (top-right corner, click your profile picture → *Settings*).
   - On the left sidebar, scroll down to **Developer settings**.
   - Click **Personal access tokens → Tokens (classic)**.
   - Click **Generate new token → Generate new token (classic)**.
     - (You can also use "Fine-grained tokens", but for Jenkins the classic ones are simpler.)

   Configure the token
   - **Note**: Give it a name like *Jenkins Pipeline Token*.
   - **Expiration**: Choose a duration (90 days, 1 year, or "No expiration" if it's a private lab).
   - **Scopes to check** (minimum for Jenkins CI/CD):
     - ☑ repo (Full control of private repositories — needed if Jenkins will clone/push code)

- ☑ workflow (if you ever want Jenkins to trigger GitHub Actions)
- ☑ read:org (optional, if your repo is under an org and Jenkins needs to read it)

Generate and Copy token

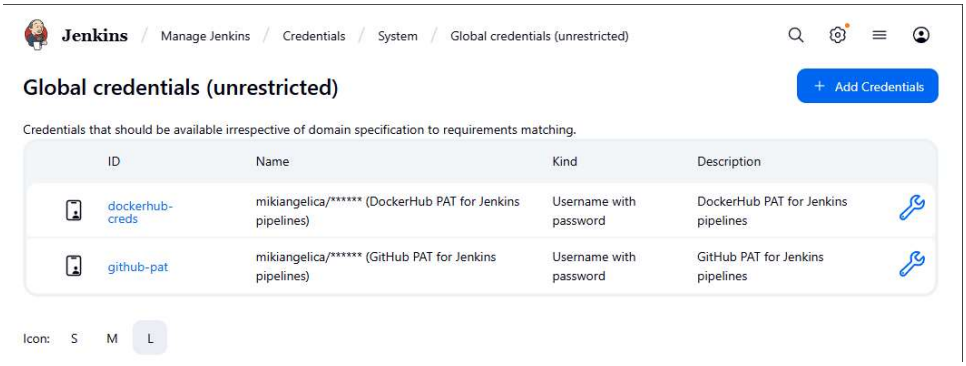
In DockerHub UI:

- Go to **DockerHub → Click on profile > Account Settings > Personal access tokens**
- Click on Generate New access token
  - Access token description: jenkins-ci
  - Expiration date: 90 days
  - Access Permissions: Read, Write, Delete

Generate and Copy token
dckr_pat_AnjFz8Qj6fQtssHvYcwAVoaQ_5c

In Jenkins,
I add credentials
Manage Jenkins > Credentials > (system) Global:

- **GitHub**:
  - Kind: Username with password
  - Scope: Global
  - Username: mikiangelica
  - Password: PAT
  - ID: github-pat
  - Description: GitHub PAT for Jenkins pipelines
- **Docker Hub**:
  - Kind: Username with password
  - Scope: Global
  - Username: mikiangelica
  - Password: PAT
  - ID: dockerhub-creds
  - Description: DockerHub PAT for Jenkins pipelines



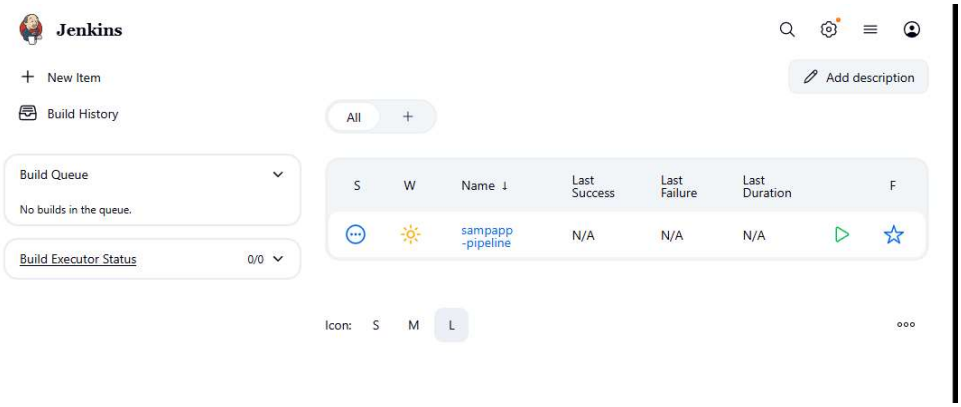
2. **Create a new pipeline job in Jenkins**
   a. Go to Jenkins Dashboard > New Item
   b. Enter name : sampapp-pipeline
   c. Choose Pipeline and click OK

Configure SCM (GitHub Repo)

Inside the job:

a. Scroll to **Pipeline** section.
b. Select **Pipeline script from SCM**.
c. SCM → Git.
d. Enter your **Repository URL** (e.g., https://github.com/mikiangelica/sampapp.git).
e. Credentials → choose the GitHub PAT credentials you added.
f. Branch Specifier → */master (since your branch is master).

This tells Jenkins to pull your code + manifests from GitHub.



Back up Cluster

```
# Base resources
$ kubectl get all --all-namespaces -o yaml > base/all.yaml

# Storage
$ k get sc,pv,pvc -Ao yaml > ../storage/storage.yaml

# Configs and secrets
$ k get cm,secrets -Aoyaml > configs.yaml

# Netpols
$ k get netpol -Aoyaml > netpols/netpols.yaml

# Apps
$ kubectl get applications.argoproj.io -A -o yaml > argocd/apps.yaml

# Appprojects
$ kubectl get appprojects.argoproj.io -A -o yaml > argocd/appprojects.yaml

# CRDs
$ kubectl get crd -o yaml > crds/all-crds.yaml

# NS
$ k get ns -oyaml > base/namespace.yaml

# SA, Roles, Rolebindings, ClusterRoles, ClusterRoleBindings
kubectl get sa,role,rolebinding,clusterrole,clusterrolebinding -A -o yaml > rbac/rbac.yaml

# Ingress
$ kubectl get ingress -A -o yaml > networking/ingresses.yaml
```

Backup to github.

Add Jenkinfile to repo root (same level as /manifests, Dockerfiles/ etc)

```
$ cat Jenkinsfile
pipeline {
  agent any
  environment {
    IMAGE_NAME = "mikiangelica/sampapp"
    IMAGE_TAG  = "v1"
  }

  stages {
    stage('Checkout Code') {
      steps {
        git branch: 'master',
          credentialsId: 'github-pat',
          url: 'https://github.com/mikiangelica/k8s-manifests.git'
      }
    }

    stage('Build & Push Docker Image') {
      steps {
        withCredentials([usernamePassword(credentialsId: 'dockerhub-creds',
                                          usernameVariable: 'DOCKER_USER',
                                          passwordVariable: 'DOCKER_PASS')]) {
          sh """
            docker build -t $IMAGE_NAME:$IMAGE_TAG -f Dockerfiles/Dockerfile .
            echo $DOCKER_PASS | docker login -u $DOCKER_USER --password-stdin
            docker push $IMAGE_NAME:$IMAGE_TAG
          """
        }
      }
    }
    stage('Update Manifests') {
      steps {
        sh """
          sed -i 's|image: .*|image: $IMAGE_NAME:$IMAGE_TAG|'
manifests/base/deployment.yaml
          git config user.email "mikiangelica@gmail.com"
          git config user.name "admin"
          git add manifests/base/deployment.yaml
          git commit -m "Update image to $IMAGE_NAME:$IMAGE_TAG" || echo "No changes
to commit"
          git push origin master
        """
      }
    }

    stage('Trigger ArgoCD Sync') {
      steps {
        withCredentials([string(credentialsId: 'argocd-token', variable: 'ARGOCD_TOKEN')]) {
```

```
            sh """
              curl -k -H "Authorization: Bearer $ARGOCD_TOKEN" \
                -X POST https://192.168.81.128:31805/api/v1/applications/sampapp/sync
            """
          }
        }
      }
    }
}
```

Take note of the following:

- dockerhub-creds → with the ID of your DockerHub credential in Jenkins.
- github-pat → with the ID of your GitHub credential in Jenkins.
- your-dockerhub-username/sampapp → your DockerHub repo.
- <ARGOCD_SERVER> → your ArgoCD API server (e.g. argocd-server.argocd.svc.cluster.local:443 or NodePort IP). - how I access UI basically
- <ARGOCD_TOKEN> → you'll need to generate a token (argocd account generate-token).

Get argocd server IP

```
$ k get svc -n argocd argocd-server
NAME          TYPE      CLUSTER-IP     EXTERNAL-IP   PORT(S)                  AGE
argocd-server   NodePort   10.43.157.58   <none>       80:31805/TCP,443:32619/TCP   4d
```

- **ClusterIP:** 10.43.157.58 → only reachable *inside* the cluster.
- **NodePort mapping:**
  - Port 80 is exposed at **NodePort 31805**
  - Port 443 is exposed at **NodePort 32619**

Get argocd token

By default the **admin account in ArgoCD does not allow API key/token generation**, unless you explicitly enable it in the config. For the purpose of this lab we will use admin account. Normally for actual setup we wld have a dedicated Jenkins Service account.

- Edit argocd-cm config map so that admin account can generate token

  ```
  # kubectl edit configmap argocd-cm -n argocd
  ```

  Under data:, add:

  ```
  data:
    accounts.admin: apiKey
  ```

- Restart deploy

  ```
  # kubectl rollout restart deployment argocd-server -n argocd
  ```

- Go to ArgoCD UI > Settings > Accounts > Admin > Generate New (token)

Created new Secret Text credential on Jenkins with above token called argocd-token

Create deployment.yaml specifically for my app since I removed all initial files and my current app manifest is in all.yaml.

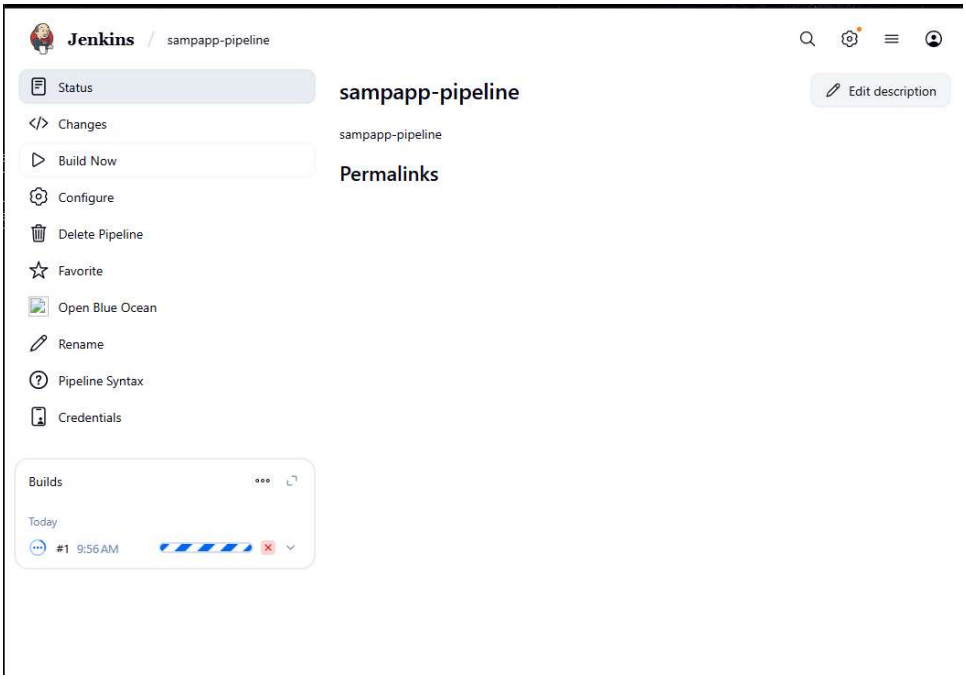
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sampapp
  namespace: default
spec:
```

```
replicas: 1
selector:
  matchLabels:
    app: sampapp
template:
  metadata:
    labels:
      app: sampapp
  spec:
    containers:
    - name: sampapp
      image: mikiangelica/sampapp:v1   # <-- Jenkins will update this line
      ports:
      - containerPort: 8080
```

Push above to Git.

Test the Pipeline in Jenkins

Go to Jenkins Job > "Build Now"



- Watch the stages: Checkout → Build → Push → Update Manifests → Trigger ArgoCD.
- Confirm on GitHub that deployment.yaml gets updated with the new tag.
- Confirm in ArgoCD UI that the app syncs and your pods pull the new image from DockerHub.

#1 (Sep 25, 2025, 9:56:49 AM) Add description Keep this build forever Started 2 min 27 sec ago Took 1 min 50 sec Started by user Jenkins Admin This run spent: 1 min 31 sec waiting; 1 min 50 sec build duration; 1 min 50 sec total from scheduled to completion. Revision: f8bf15ec15040f22b06a4f90a63c59253e509269 Repository: https://github.com/mikiangelica/sampapp.git refs/remotes/origin/master Revision: f8bf15ec15040f22b06a4f90a63c59253e509269 Repository: https://github.com/mikiangelica/k8s-manifests.git refs/remotes/origin/master The following steps that have been detected may have insecure interpolation of sensitive variables (click here for an explanation): sh: [DOCKER_PASS] No changes.

ISSUE 4: BUILD FAILED 1 (same tag version)
CAUSE: Current deployment.yaml and image tag in DockerHub are same - v1
FIX: Changed deployment.yaml's image tag to oldtag instead of v1
Troubleshooting Steps Overview: Checked console output of build in Jenkins. Changed deployment.yaml's image tag to oldtag instead of v1.
Reference: Troubleshooting Log 4

ISSUE 5: BUILD FAILED 2 (Indentation Error)
CAUSE: Indentation error in Jenkinsfile
FIX: Added 4 tabs in Jenkinsfile before image to handle indentation
Troubleshooting Steps Overview: Checked console output of build in Jenkins. Added 4 tabs in Jenkinsfile before image to handle indentation
Reference: Troubleshooting Log 4

ISSUE 6: BUILD FAILED 3 (Docker not found)
CAUSE: Docker not found bc jenkins agent does not by default include Docker
FIX: Made jenkins-agent file for jenkins to use
Troubleshooting Steps Overview: Checked console output of build in Jenkins. Made jenkins-agent file for jenkins to use with docker. Registered this custom agent in Jenkins UI and Jenkinfile.
Reference: Troubleshooting Log 4

ISSUE 7: BUILD FAILED 4 (Didn't push above Jenkinsfile changes to GitHub - silly)
CAUSE: Jenkins detects from GitHub and I didn't push it there!
FIX: Pushed image
Troubleshooting Steps Overview: I realised it after I pressed Build Now button for the 4th time...
Reference: Troubleshooting Log 4

ISSUE 8: BUILD FAILED 5 (Could not find user)
CAUSE: Amended user in Dockerfile.jenkins-agent
FIX: Amended user in Dockerfile.jenkins-agent from admin to jenkins
Troubleshooting Steps Overview: Checked console output of build in Jenkins. Then amended Dockerfile.jenkins-agent USER at end of file. Repushed Dockerfile to DockerHub
Reference: Troubleshooting Log 4

ISSUE 9: BUILD FAILED 6 (No docker daemon)
CAUSE: No Docker daemon running inside the Jenkins agent container
FIX: Added hostpath volume to jenkins pod template for jenkins agent (hosts' docker daemon socket file)
Troubleshooting Steps Overview: Checked console output of build in Jenkins. Added hostpath volume to jenkins pod template for jenkins agent (hosts' docker daemon socket file)
Reference: Troubleshooting Log 4

ISSUE 10: BUILD FAILED 7 (Permission denied error)
CAUSE: Usually the above socket file is owned by root and group docker on the host.
But in our container, Jenkins run as user jenkins (not root, and not in docker group)
So when docker CLI tries to talk to the socket, the kernel blocks it > permission denied.
FIX: Removed last USER jenkins line from Dockerfile.jenkins-agent-build
Troubleshooting Steps Overview: Checked console output of build in Jenkins. Removed last USER jenkins line from Dockerfile.jenkins-agent-build and repushed to DockerHub

Reference: Troubleshooting Log 4

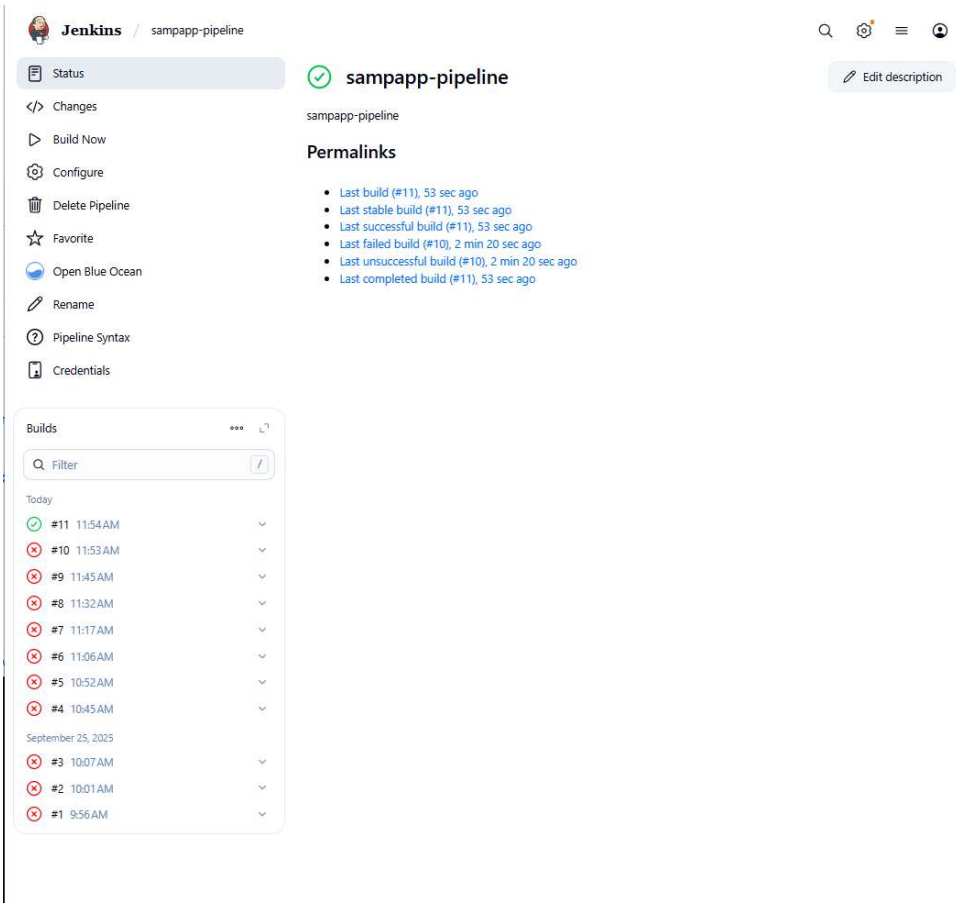
ISSUE 11: BUILD FAILED 8
CAUSE: Wrong path in application Dockerfile (because I had moved my Dockerfile to a subdir after creating above custom jenkins agent Dockerfile
FIX: Amended path to my sampapp Dockerfile
Troubleshooting Steps Overview: Checked console output of build in Jenkins. Amended path to my sampapp Dockerfile
Reference: Troubleshooting Log 4

ISSUE 11: BUILD FAILED 8 (Dockerfile: no such file or dir)
CAUSE: Wrong path in application Dockerfile (because I had moved my Dockerfile to a subdir after creating above custom jenkins agent Dockerfile
FIX: Amended path to my sampapp Dockerfile
Troubleshooting Steps Overview: Checked console output of build in Jenkins. Amended path to my sampapp Dockerfile
Reference: Troubleshooting Log 4

ISSUE 12: BUILD FAILED 8 (Dockerfile: no such file or dir)
CAUSE: No github credentials in Jenkinsfile
FIX: Added github credentials in Jenkinsfile (github PAT)
Troubleshooting Steps Overview: Checked console output of build in Jenkins. Added github credentials in Jenkinsfile (github PAT)
Reference: Troubleshooting Log 4

Repushed Jenkinsfile to Github. Had to push twice more (because 10th try I missed out 1 } in the Jenkinsfile which caused it to fail)

Finally 11th time's the charm! It succeeded….



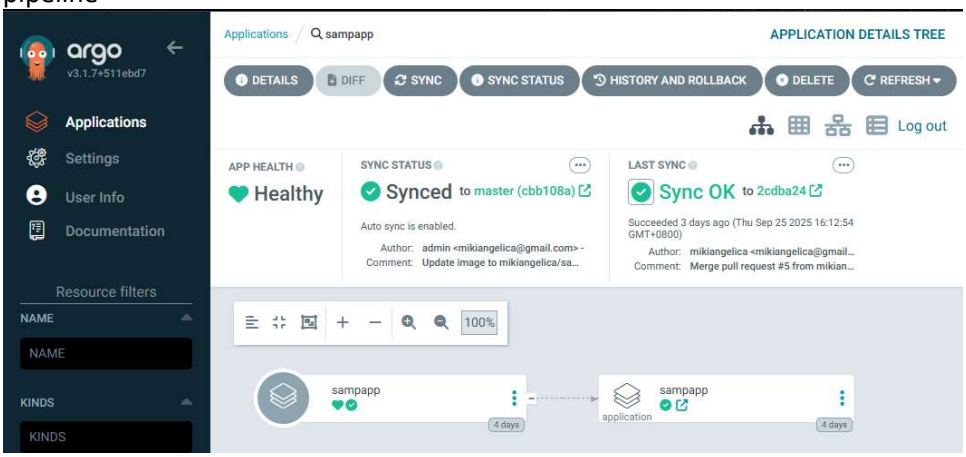
$ k get app -n argocd sampapp
NAME SYNC STATUS HEALTH STATUS
sampapp Synced Healthy

So what did this pipeline do exactly?

1. Checkout code
    a. Jenkins checked out 2 repos (sampapp and k8s-manifests)
2. Build & Pushed Docker Images
    a. Inside our custom docker-build agent (with Docker CI + buildx and mounted Docker socket):
        i. Ran docker build -t mikiangelica/sampapp:v1 -f Dockerfiles/sampapp/Dockerfile . >> Built a new Docker image of app
        ii. Logged into Docker Hub using Jenkins-stored credentials (dockerhub-creds).
        iii. Pushed the new image >> docker.io/mikiangelica/sampapp:v1
3. Update Manifests
    a. Edited manifests/base/deployment.yaml with sed to replace the old iamge with the new one (mikiangelica/sampapp:v1)
    b. Committed the change back to the k8s-manifests repo using Github PAT.
    c. Pushed the commit > updated repo now contains the new image ref
    d. GitOps source of truth (manifests repo) points to the latest image
4. Trigger ArgoCD Sync
    a. Jenkins called the ArgoCD API with the bearer token to sync the sampapp app to match the latest manifest in the k8s-manifests repo.

One thing to note here:
In ArgoCD UI we can see that the last sync was 3 days ago even though we just successfully ran the

pipeline



We also see that the pods in the cluster did not get redeployed/restarted:

default 7d4h	sampapp-f6c65db94-l72nz	2/2	Running	6 (113m ago)
default 7d4h	sampapp-f6c65db94-wwfg4	2/2	Running	6 (113m ago)

Why?
Because even though we changed the deployment.yaml to image:oldtag and made it so that Jenkins wld update the image to v1, there weren't actually any real changes to the app source code.
And so, ArgoCD sees that there is nothing new to apply hence there was no redeployment.

And the actual pods were already labelled v1 actually (they were deployed with v1 before we changed to oldtag), so no manifest diff > no new commit > no Pod restart
K8s Deployments only roll Pods if the image tag changed fr (or if you forced a rollout).

So the pipeline did work but just that ArgoCD saw that there wasn't actually anything to update.

**Can consider trying to resync pipeline with new manifest changes and image tag

18 Ansible to provision a new node

Saturday, 13 September 2025 11:54 pm

Technology:
Ansible (Infrastructure Automation)

1. **Set up Ansible VM**

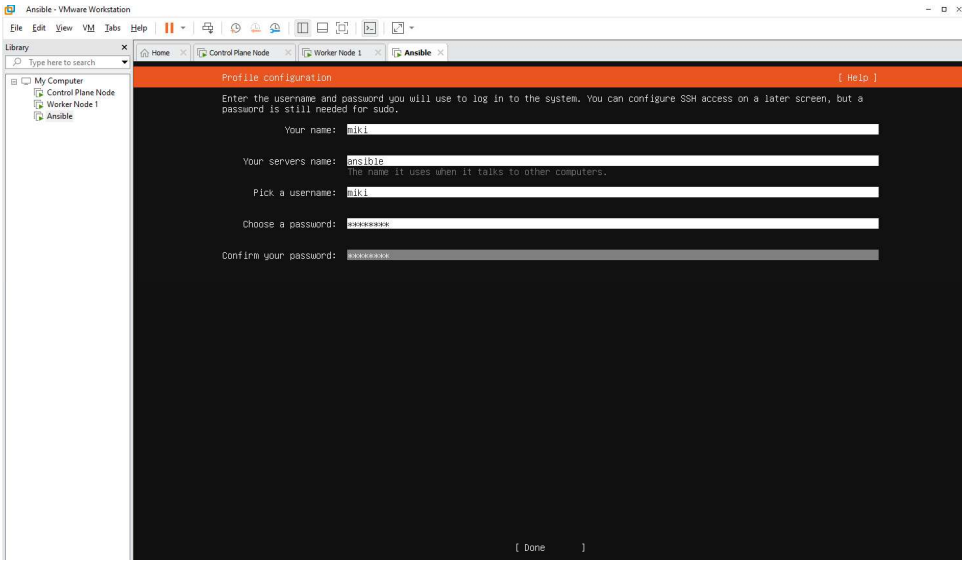
My computer is Windows and Ansible doesn't run natively on Windows. Either we use WSL or spin up a Linux VM for Ansible. Let's do the latter!

I gave 2CPU, 1GB RAM and 10GB Disk.

1. Click on "Create a New Virtual Machine"
2. Select Installer disc image file (iso)
   - Click on Browse and navigate to the directory I saved the ISO Image:
   - *C:\Users\mikia\Downloads\ubuntu-22.04.5-live-server-amd64.iso*
3. Name VM (Ansible)
4. Select "Store virtual disk as a single file"
5. Customize Hardware > New CD/DVD (SATA) change to Auto detect
6. Click Finish and let OS boot up

2. **System Bootup**

a) Click on first option in GRUB menu (Try or install Ubuntu Server)
b) Keep clicking Done for all the options offered
c) Ensure IP is on same L2 network as k3s nodes
d) Name the server as your name, nodename (ansible), your username and pw
e) Select Install OpenSSH server
f) Wait for configurations to complete then click Reboot Now
g) When screen shows "failed to mount /cdrom", press enter
h) Login to server and shutdown now
i) Click on VM settings and edit CD/DVD (SATA) to Use physical drive (autodetect)
j) Power on VM again



It kept hanging at the username configuration page so I gave it 2GB RAM instead.

Now boots up.

3. **Set up ssh for localhost to this ansible node**
   - Copy pasted ssh pubkey from localhost to ansible node
   - Made sure file is one liner
   - Changed ownership & permissions
   - Remove # from AuthorizedKeysFile line in sshd_config
   - Restarted ssh daemon
   - Added ansible to ~/.ssh/config file so I can ssh ansible instead of IP

4. **Set up ssh for ansible node to master1 & worker1 too**

```
# ssh-keygen -t rsa -b 4096 -C "ansible-control" -N "" -f ~/.ssh/id_rsa
Generating public/private rsa key pair.
Your identification has been saved in /home/miki/.ssh/id_rsa
Your public key has been saved in /home/miki/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:uHw1xjRtAheXoUROOBd3SuqHannyfo0gAM7gX6k2pc0 ansible-control
The key's randomart image is:
+---[RSA 4096]----+
|       .oBo=o.  |
|  . .   o*.B.o  |
| . + . . oB +   |
|  . o =. + =    |
|   . B..S B .   |
|    *.E..=.o    |
|   . .o *... o  |
|      o +  o .  |
|       .o.      |
+----[SHA256]-----+
```

Ansible-control pubkey

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAACAQDKmEJEqziRlRpsbdblOkC6AeU/mEKV8Lw9ewfM2kNKaf3o1jQR
```

```
wF4v2bjq94WY3XSOrPgwv22DF1PKSxFWlUv44Gx1oOLYSoNEWNt+HmS1sVmHCiLm5HVO641/ntopNL
WRPnqIa10msrZgLWvbjnW6
+j9gAR1b23Bvl9JyRKz897rpN+reksQCeTZnsBbcjdJX35Ihlq71YL8Eo7Uifglb/3aeRSjk/Nh8/N/UpY1sJGJTn/
76JJMRoGE6oAiA1dNtsVilZFao/5K4Hj7quYURqTrrkVskIt/a0rZLGdMQuYJtgLhZqBP7oBRiu63yoHQOa7qL
S3ARqrXjWqn7FPgTFo3B0s2qSFPRiBzAzLibOH+XVj3SPsKLI+JYOR9auVsNOlxEOtT5KVfO+u2Ui4pb/InCOt
7sXo9xCtK3rOuUev9cZbw4m1QgBqhkY+AD1cOOlt3TsyDxfB7qsIkEVT+J+oqDcrTV1WsITySJdyggEgJQQc0
TatfYdYHbveczq1NA0XMwRoPYK3Rom69Qn4VoKhqjgbXisZ3LeJmEe+HuwfR6Fh5dsLZ9LhrEgRd4qCD4py
Tk8pF4vBHDplqMIgdg4p/LX5ZLc14PzSrYEIzCWkZ6AhY2hUo45wHCDWxYgm2vWWU3FD/rb89bRVBbA4
cH5hKzjf7UDLxajlpnFNZYgQ== ansible-control
```

Make one liner

```
sed -i 's/\r$//' /home/miki/.ssh/authorized_keys
```

Created the ~/.ssh/config file to ssh via hostname

```
# Master node
Host master1
      HostName 192.168.81.128
      User miki
      IdentityFile ~/.ssh/id_rsa

# Worker node 1
Host worker1
      HostName 192.168.81.129
      User miki
      IdentityFile ~/.ssh/id_rsa

# Worker node 2
Host worker2
      HostName 192.168.81.134
      User miki
      IdentityFile ~/.ssh/id_rsa
```

5. **Installing Ansible**

Ansible only needs to run on **one machine**. It uses SSH to talk to other nodes.
For Ubuntu/Debian:

```
sudo apt update
sudo apt install -y software-properties-common
sudo add-apt-repository --yes --update ppa:ansible/ansible
sudo apt install -y ansible
```

Check version:

```
root@ansible:~# ansible --version
ansible [core 2.17.14]
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/root/.ansible/plugins/modules',
'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  ansible collection location = /root/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/bin/ansible
  python version = 3.10.12 (main, Aug 15 2025, 14:32:43) [GCC 11.4.0] (/usr/bin/python3)
  jinja version = 3.0.3
```

```
libyaml = True
```

6. **Set up worker2 VM**
   - Set up SSH between worker2 and localhost
   - Set up SSH between worker2 and Ansible

7. **Set up Ansible Inventory in the following dir (common ansible dir):**

```
#mkdir ~/ansible-k8s/
```

Place the following files in the above dir:
- inventory.ini → inventory file with your nodes
- playbook.yml → your playbooks
- group_vars/, roles/, etc. (later if you structure more complex projects)

In **inventory.ini** :

```
[k8s_masters]
master1 ansible_host=192.168.81.128 ansible_user=miki

[k8s_workers]
worker1 ansible_host=192.168.81.129 ansible_user=miki
worker2 ansible_host=192.168.81.134 ansible_user=miki
```

Explanation:
- [k8s_masters] = group name.
- master1 = alias for your node (can be anything meaningful).
- ansible_host=192.168.81.128 = the actual IP address of the node.
- ansible_user=ubuntu = the SSH user you log in with.

8. **Check connectivity:**

ansible all -i inventory.ini -m ping
☑ You should see pong back from each host.

```
miki@ansible:~/ansible-k8s$ ansible all -i inventory.ini -m ping
[WARNING]: Platform linux on host worker2 is using the discovered Python interpreter at
/usr/bin/python3.10, but future installation of another Python interpreter
could change the meaning of that path. See https://docs.ansible.com/ansible-
core/2.17/reference_appendices/interpreter_discovery.html for more information.
worker2 | SUCCESS => {
   "ansible_facts": {
      "discovered_interpreter_python": "/usr/bin/python3.10"
   },
   "changed": false,
   "ping": "pong"
}
[WARNING]: Platform linux on host master1 is using the discovered Python interpreter at
/usr/bin/python3.10, but future installation of another Python interpreter
could change the meaning of that path. See https://docs.ansible.com/ansible-
core/2.17/reference_appendices/interpreter_discovery.html for more information.
master1 | SUCCESS => {
   "ansible_facts": {
```

```
        "discovered_interpreter_python": "/usr/bin/python3.10"
    },
    "changed": false,
    "ping": "pong"
}
[WARNING]: Platform linux on host worker1 is using the discovered Python interpreter at
/usr/bin/python3.10, but future installation of another Python interpreter
could change the meaning of that path. See https://docs.ansible.com/ansible-
core/2.17/reference_appendices/interpreter_discovery.html for more information.
worker1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3.10"
    },
    "changed": false,
    "ping": "pong"
}
```

In the above, Ansible read the inventory.ini file and saw master1 & worker1. Then it used SSH to connect to each host, discovered the python interpreter on each node, Ran the ping module which tests connection & python availability. Both returned pong means Ansible can successfully connect, authenticate, and execute commands on both nodes! :D

Test something slightly more advanced:

```
miki@ansible:~/ansible-k8s$ ansible all -i inventory.ini -m shell -a "hostname && uptime"
[WARNING]: Platform linux on host worker1 is using the discovered Python interpreter at
/usr/bin/python3.10, but future installation of another Python interpreter
could change the meaning of that path. See https://docs.ansible.com/ansible-
core/2.17/reference_appendices/interpreter_discovery.html for more information.
worker1 | CHANGED | rc=0 >>
worker1
 17:53:34 up  1:28,  1 user,  load average: 0.60, 0.28, 0.20
[WARNING]: Platform linux on host worker2 is using the discovered Python interpreter at
/usr/bin/python3.10, but future installation of another Python interpreter
could change the meaning of that path. See https://docs.ansible.com/ansible-
core/2.17/reference_appendices/interpreter_discovery.html for more information.
worker2 | CHANGED | rc=0 >>
worker2
 17:53:34 up 4 min,  1 user,  load average: 0.02, 0.07, 0.03
[WARNING]: Platform linux on host master1 is using the discovered Python interpreter at
/usr/bin/python3.10, but future installation of another Python interpreter
could change the meaning of that path. See https://docs.ansible.com/ansible-
core/2.17/reference_appendices/interpreter_discovery.html for more information.
master1 | CHANGED | rc=0 >>
master1
 17:53:34 up  1:29,  1 user,  load average: 0.06, 0.04, 0.08
```

9. Create folder structure

```
ansible-k8s/
├── inventory/
│   └── inventory.ini
```

```
├── playbooks/
│   └── add-worker.yml
└── roles/
    ├── common/
    │   └── tasks/main.yml
    └── worker/
        └── tasks/main.yml
```

10. Create the main playbook

```
---
- name: Configure and join new Kubernetes worker node
  hosts: k8s_workers
  become: true
  vars:
    k3s_token:
"K1080e8d3f7b27ad61ef4ca4d60c0f53705c52f3aeb3c17a4d782cedcdd865c1430::server:60f6d6681f7e
54098d82c7f01d32f64b"
    master_ip: "192.168.81.128"
  roles:
    - common
    - worker
```

Find node token on master1:

```
sudo cat /var/lib/rancher/k3s/server/node-token
```

Create common role task main playbook (~/ansible-k8s/roles/common/tasks/main.yml)

```
---
- name: Update apt packages
  apt:
    update_cache: yes
    upgrade: yes

- name: Install dependencies
  apt:
    name:
      - curl
      - apt-transport-https
      - ca-certificates
      - software-properties-common
    state: present
```

Create worker role task main playbook (~/ansible-k8s/roles/worker/tasks/main.yml)

```
---
- name: Install K3s agent (worker)
  shell: |
    curl -sfL https://get.k3s.io | K3S_URL=https://{{ master_ip }}:6443 K3S_TOKEN={{ k3s_token }} sh -
  args:
    creates: /etc/systemd/system/k3s-agent.service
  register: k3s_agent_install
```

```
- name: Show K3s installation output
  debug:
    var: k3s_agent_install.stdout
```

11. Run the playbook

To run playbook

```
cd ~/ansible-k8s
ansible-playbook -i inventory/inventory.ini playbooks/add-worker.yaml
```

Ran into issues

ISSUE 2: Playbook not running because role not found
CAUSE: roles dir in wrong path
FIX: Moved roles dir to under playbooks dir
Troubleshooting Steps Overview: Saw error upon trying to run playbook to add worker2 into cluster is related to role 'common' not being found, moved roles dir to path and looks like playbook can detect it now
Reference: Troubleshooting Log 1

ISSUE 3: Playbook not running because passwordless sudo not configured for user
CAUSE: ansible user does not have passwordless sudo
FIX: Added passwordless sudo for ansible user
Troubleshooting Steps Overview: Error during playbook run states Missing sudo password, added user to visudo file for passwordless sudo
Reference: Troubleshooting Log 1

Run playbook:

```
miki@ansible:~/ansible-k8s$ ansible-playbook -i inventory/inventory.ini playbooks/add-worker.yaml

PLAY [Configure and join new Kubernetes worker node]
**********************************************************************************
*****************************

TASK [Gathering Facts]
**********************************************************************************
************************************************************
[WARNING]: Platform linux on host worker2 is using the discovered Python interpreter at
/usr/bin/python3.10, but future installation of another Python interpreter
could change the meaning of that path. See https://docs.ansible.com/ansible-
core/2.17/reference_appendices/interpreter_discovery.html for more information.
ok: [worker2]
[WARNING]: Platform linux on host worker1 is using the discovered Python interpreter at
/usr/bin/python3.10, but future installation of another Python interpreter
could change the meaning of that path. See https://docs.ansible.com/ansible-
core/2.17/reference_appendices/interpreter_discovery.html for more information.
ok: [worker1]

TASK [common : Update apt packages]
**********************************************************************************
*******************************************
changed: [worker2]
```

```
changed: [worker1]

TASK [common : Install dependencies]
*********************************************************************************************
*****************************************
changed: [worker2]
changed: [worker1]

TASK [worker : Install K3s agent (worker)]
*********************************************************************************************
************************************
ok: [worker1]
changed: [worker2]

TASK [worker : Show K3s installation output]
*********************************************************************************************
************************************
ok: [worker1] => {
   "k3s_agent_install.stdout": "skipped, since /etc/systemd/system/k3s-agent.service exists"
}
ok: [worker2] => {
   "k3s_agent_install.stdout": "[INFO]  Finding release for channel stable\n[INFO]  Using v1.33.5+k3s1
as release\n[INFO]  Downloading hash https://github.com/k3s-io/k3s/releases/download/v1.33.5
+k3s1/sha256sum-amd64.txt\n[INFO]  Downloading binary https://github.com/k3s-
io/k3s/releases/download/v1.33.5+k3s1/k3s\n[INFO]  Verifying binary download\n[INFO]  Installing k3s
to /usr/local/bin/k3s\n[INFO]  Skipping installation of SELinux RPM\n[INFO]  Creating
/usr/local/bin/kubectl symlink to k3s\n[INFO]  Creating /usr/local/bin/crictl symlink to k3s\n[INFO]
Creating /usr/local/bin/ctr symlink to k3s\n[INFO]  Creating killall script /usr/local/bin/k3s-killall.sh
\n[INFO]  Creating uninstall script /usr/local/bin/k3s-agent-uninstall.sh\n[INFO]  env: Creating
environment file /etc/systemd/system/k3s-agent.service.env\n[INFO]  systemd: Creating service file
/etc/systemd/system/k3s-agent.service\n[INFO]  systemd: Enabling k3s-agent unit\n[INFO]  systemd:
Starting k3s-agent"
}

PLAY RECAP
*********************************************************************************************
*****************************************************************
worker1                    : ok=5    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
worker2                    : ok=5    changed=3    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Verify:

```
$ kubectl get no -owide
NAME     STATUS   ROLES           AGE   VERSION       INTERNAL-IP     EXTERNAL-IP   OS-IMAGE
KERNEL-VERSION      CONTAINER-RUNTIME
master1   Ready    control-plane,master   26d   v1.33.4+k3s1   192.168.81.128   <none>      Ubuntu
22.04.5 LTS   5.15.0-156-generic   containerd://2.0.5-k3s2
worker1   Ready    <none>          26d   v1.33.4+k3s1   192.168.81.129   <none>      Ubuntu 22.04.5
LTS   5.15.0-156-generic   containerd://2.0.5-k3s2
worker2   Ready    <none>          84s   v1.33.5+k3s1   192.168.81.134   <none>      Ubuntu 22.04.5
LTS   5.15.0-157-generic   containerd://2.1.4-k3s1
```

SUCCESS!

19 Istio ServiceMesh

Saturday, 13 September 2025 11:55 pm

Technology:
Istio ServiceMesh

1. **Confirm cluster access**

```
root@master1:~# kubectl get no
NAME     STATUS   ROLES                  AGE    VERSION
master1  Ready    control-plane,master   26d    v1.33.4+k3s1
worker1  Ready    <none>                 26d    v1.33.4+k3s1
worker2  Ready    <none>                 8m16s  v1.33.5+k3s1
```

2. **Install istioctl locally**
   - Localhost already has kubeconfig with full admin access to cluster
   - Can version-control and rerun istio installs easily from there
   - Avoid messing with control plane environment

```
# mkdir -p ~/istio && cd ~/istio
# curl -kLO https://github.com/istio/istio/releases/download/1.27.1/istio-1.27.1-win.zip
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100 29.1M  100 29.1M    0     0  4861k      0  0:00:06  0:00:06 --:--:-- 5860k
# unzip istio-1.27.1-win.zip -d istio-1.27.1
# export PATH=$HOME/istio/istio-1.27.1/istio-1.27.1/bin:$PATH
# istioctl version
Istio is not present in the cluster: no running Istio pods in namespace "istio-system"
client version: 1.27.1
```

   - Add istioctl path to environment variables (system properties) so it is persistent

3. **Install the Istio Control Plane (Default Profile)**

```
# istioctl install --set profile=demo -y
     |\
     | \
     |  \
     |   \
    /||    \
   / ||     \
  /  ||      \
 /   ||       \
/_____||_______\
_________________
 \__     _____/
```

```
   \_____/

✔ Istio core installed
✔ Istiod installed
✔ Ingress gateways installed
✔ Egress gateways installed
✔ Installation complete
```

demo = a full-featured profile (Ingress, Egress, Prometheus, etc.)
Perfect for learning and testing.

It'll create:

- Namespace istio-system
- Control plane pods (istiod, istio-ingressgateway, etc.)
- Webhooks, CRDs, etc.

Verify pods are there now:

```
# k get po -n istio-system
NAME                                    READY   STATUS    RESTARTS   AGE
istio-egressgateway-74bbd4975d-5xftt    1/1     Running   0          80s
istio-ingressgateway-7c5fd9bc88-zq95g   1/1     Running   0          80s
istiod-597958d8ff-x64wr                 1/1     Running   0          116s
```

4. **Label application namespace**

To enable automatic sidecar injection (Envoy proxy)
Check which ns the app is deployed in

```
# kubectl get po -A | grep sampapp
default          sampapp-f6c65db94-l72nz                  2/2     Running         14
(3h5m ago)   13d
default          sampapp-f6c65db94-wwfg4                   2/2     Running         14
(3h5m ago)   13d
```

```
# kubectl label namespace default istio-injection=enabled
namespace/default labeled
```

5. **Restart app deployment**

```
# k rollout restart deployment sampapp -n default
deployment.apps/sampapp restarted
```

Wait a couple mins and you'll see that sampapp has 3 containers now instead of 2

```
sampapp-cfdc49c4b-bddhl    3/3    Running    0          113s
sampapp-cfdc49c4b-f9knr    3/3    Running    0          3m
```

```
$ k describe po sampapp-cfdc49c4b-bddhl -n default | grep -i "image:"
   Image:        docker.io/istio/proxyv2:1.27.1
   Image:        docker.io/istio/proxyv2:1.27.1
   Image:         mikiangelica/sampapp:v1
   Image:        nicolaka/netshoot
```

6. **Verify that app is part of the mesh**

```
# istioctl proxy-status
NAME                                                   CLUSTER       ISTIOD                  VERSION
SUBSCRIBED TYPES
istio-egressgateway-74bbd4975d-5xftt.istio-system      Kubernetes
istiod-597958d8ff-x64wr     1.27.1     3 (CDS,LDS,EDS)
istio-ingressgateway-7c5fd9bc88-zq95g.istio-system     Kubernetes
istiod-597958d8ff-x64wr     1.27.1     3 (CDS,LDS,EDS)
sampapp-cfdc49c4b-bddhl.default                        Kubernetes     istiod-597958d8ff-
x64wr     1.27.1     4 (CDS,LDS,EDS,RDS)
sampapp-cfdc49c4b-f9knr.default                        Kubernetes     istiod-597958d8ff-
x64wr     1.27.1     4 (CDS,LDS,EDS,RDS)
```

Awesome it worked.

7. **Check app service**

```
# k get svc -n default sampapp-service
NAME             TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
sampapp-service   ClusterIP   10.43.133.187   <none>        5000/TCP   26d
```

8. **Create Istio Gateway manifest**

   In manifests/istio/sampapp-gateway.yaml

```
apiVersion: gateway.networking.k8s.io/v1
kind: Gateway
metadata:
  name: sampapp-gateway
  namespace: default
spec:
  gatewayClassName: istio
  listeners:
  - name: http
    protocol: HTTP
    port: 80
    allowedRoutes:
      namespaces:
        from: Same
```

```
# k apply -f sampapp-gateway.yaml
gateway.networking.istio.io/sampapp-gateway created
```

9. **Create VirtualService**

   In manifests/istio/sampapp-virtualservice.yaml

```
apiVersion: networking.istio.io/v1beta1
kind: VirtualService
metadata:
  name: sampapp
  namespace: default
spec:
  hosts:
  - "*"
```

```
 gateways:
 - sampapp-gateway
 http:
 - match:
   - uri:
       prefix: /
   route:
   - destination:
       host: sampapp-service   # match svc
       port:
         number: 5000    # match svc
```

```
# k apply -f sampapp-virtualservice.yaml
virtualservice.networking.istio.io/sampapp created
```

10. Check that both are up

```
# k get gateway,vs
NAME                                    CLASS   ADDRESS   PROGRAMMED   AGE
gateway.gateway.networking.k8s.io/sampapp-gateway   istio           False       66s

NAME                                  GATEWAYS           HOSTS   AGE
virtualservice.networking.istio.io/sampapp   ["sampapp-gateway"]   ["*"]   24m
```

Note that for gateway, PROGRAMMED is False and there is no Address as it is not yet bound to an istio ingress gateway service.

11. Link to istio ingress gateway svc

Verify istio ingress gateway class

```
# k get gatewayclass
NAME         CONTROLLER                ACCEPTED   AGE
istio        istio.io/gateway-controller   True       51m
istio-remote   istio.io/unmanaged-gateway    True       51m
```

Check that istio ingress gateway svc exists

```
# k get svc -n istio-system
NAME                   TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)
AGE
istio-egressgateway    ClusterIP      10.43.143.163   <none>        80/TCP,443/TCP
52m
istio-ingressgateway   LoadBalancer   10.43.71.123    <pending>
15021:32477/TCP,80:31171/TCP,443:31707/TCP,31400:30106/TCP,15443:30688/TC
P   52m
istiod                 ClusterIP      10.43.73.122    <none>
15010/TCP,15012/TCP,443/TCP,15014/TCP                                  52m
```

Patch istio ingress gateway svc to use NodePort instead of LoadBalancer

```
# kubectl patch svc istio-ingressgateway -n istio-system -p '{"spec": {"type":
"NodePort"}}'
```

Check ingressgateway again and take note of the "80:**31171**/TCP"

```
# kubectl get svc -n istio-system istio-ingressgateway
NAME                    TYPE       CLUSTER-IP     EXTERNAL-IP   PORT(S)
AGE
istio-ingressgateway   NodePort   10.43.71.123   <none>
15021:32477/TCP,80:31171/TCP,443:31707/TCP,31400:30106/TCP,15443:30688/TC
P   53m
```

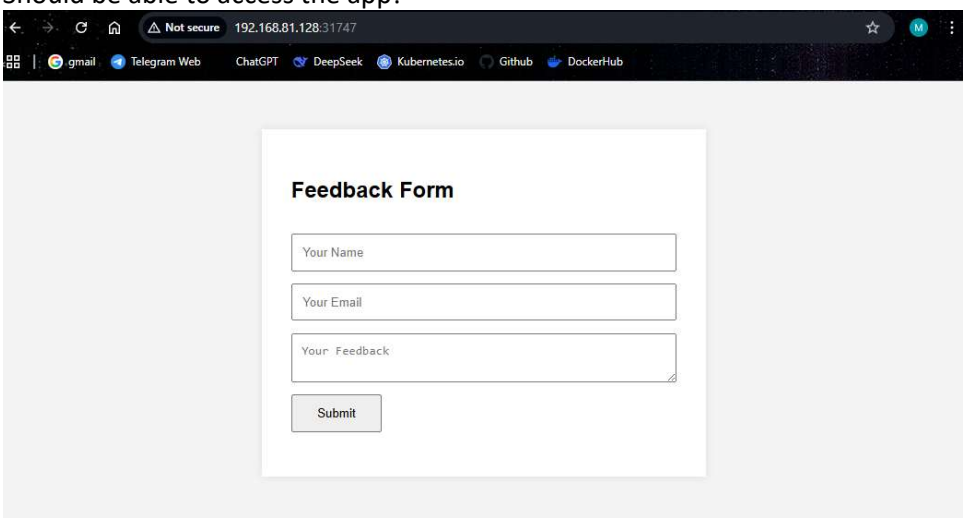
12. **Wait for gateway to become programmed**

ISSUE 4: GATEWAY NOT BECOMING PROGRAMMED "TRUE"
CAUSE: Wrong service type and VS are no longer attached to in 1.27+
FIX: Changed from loadbalancer to Nodeport, and VS to HTTPRoute
Troubleshooting Steps Overview: Checked yaml of gateway and saw error, changed from VS to HTTPRoute. Checked gateway svc type, tried to patch (failed). Deleted gateway (which owns svc) and applied new gateway yaml that states NodePort for svc
Reference: Troubleshooting Log 1

Programmed = True

```
$ k get svc sampapp-gateway-istio
NAME                    TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)                    AGE
sampapp-gateway-istio   NodePort   10.43.252.215   <none>
15021:30857/TCP,80:31747/TCP   22s

$ k get gateway
NAME              CLASS   ADDRESS                                      PROGRAMMED   AGE
sampapp-gateway   istio   sampapp-gateway-istio.default.svc.cluster.local   True         41s
```

13. **Test access to app**
    On browser, go to:
    http://192.168.81.128:31747
    Should be able to access the app!



Can use any of the nodes in clusters' IPs to access.

20 Prometheus + Grafana

Saturday, 13 September 2025 11:55 pm

Technology:
Prometheus
Grafana
Kiali

1. **Create namespace for monitoring**

```
# kubectl create namespace monitoring
```

2. **Install kube-prometheus-stack via Helm**
(includes Prometheus, Grafana, Alertmanager) - most popular setup

```
# helm repo add prometheus-community https://prometheus-
community.github.io/helm-charts

# helm repo update

# helm install prometheus prometheus-community/kube-prometheus-stack \
  --namespace monitoring \
  --set grafana.enabled=true \
  --set prometheus.prometheusSpec.serviceMonitorSelectorNilUsesHelmValues=false

NAME: prometheus
LAST DEPLOYED: Sun Oct  5 10:02:58 2025
NAMESPACE: monitoring
STATUS: deployed
REVISION: 1
NOTES:
kube-prometheus-stack has been installed. Check its status by running:
  kubectl --namespace monitoring get pods -l "release=prometheus"

Get Grafana 'admin' user password by running:

  kubectl --namespace monitoring get secrets prometheus-grafana -o
jsonpath="{.data.admin-password}" | base64 -d ; echo

Access Grafana local instance:

  export POD_NAME=$(kubectl --namespace monitoring get pod -l
"app.kubernetes.io/name=grafana,app.kubernetes.io/instance=prometheus" -oname)
  kubectl --namespace monitoring port-forward $POD_NAME 3000

Visit https://github.com/prometheus-operator/kube-prometheus for instructions on how
to create & configure Alertmanager and Prometheus instances using the Operator.
```

Verify pods are up:

```
$ k get po -n monitoring -owide
NAME                                                     READY   STATUS    RESTARTS   AGE    IP              NODE
NOMINATED NODE   READINESS GATES
alertmanager-prometheus-kube-prometheus-alertmanager-0   2/2     Running   0
109s   172.18.189.84   worker2   <none>           <none>
prometheus-grafana-674cf8cb44-hgjmf                      3/3     Running   0          2m12s
172.18.189.83   worker2   <none>           <none>
prometheus-kube-prometheus-operator-77c7674c65-8776m     1/1     Running   0
2m12s   172.18.189.82   worker2   <none>           <none>
prometheus-kube-state-metrics-7c5fb9d798-g9cn4           1/1     Running   0          2m12s
172.18.189.81   worker2   <none>           <none>
prometheus-prometheus-kube-prometheus-prometheus-0       2/2     Running   0
108s   172.18.189.85   worker2   <none>           <none>
prometheus-prometheus-node-exporter-64vtm                1/1     Running   0          2m12s
192.168.81.134   worker2   <none>           <none>
prometheus-prometheus-node-exporter-7vbl5                1/1     Running   0          2m12s
192.168.81.129   worker1   <none>           <none>
prometheus-prometheus-node-exporter-qbnpb                1/1     Running   0          2m12s
192.168.81.128   master1   <none>           <none>
```

3. **Enable istio metrics scraping**

   Istio exposes metrics automatically via istiod, istio-proxy etc.. Just need to enable Prometheus scraping annotations

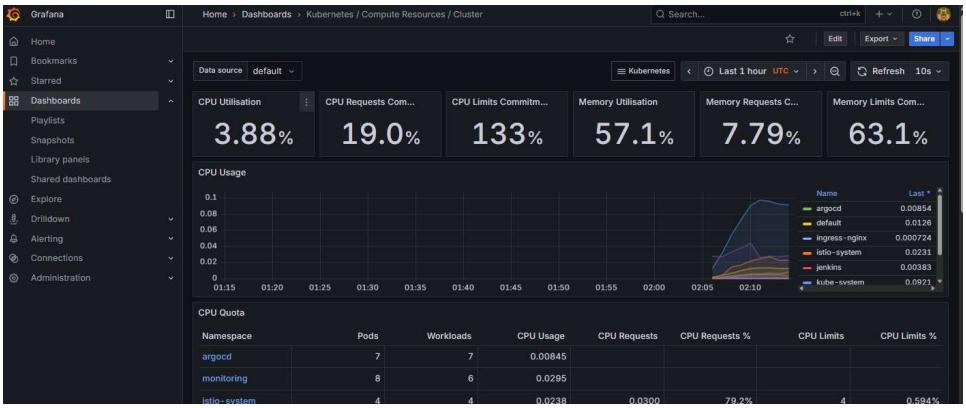
```
# k label ns istio-system istio-injection=enabled --overwrite
namespace/istio-system labeled

# kubectl apply -f
https://raw.githubusercontent.com/istio/istio/release-1.27/samples/addons/prometheus.yaml
serviceaccount/prometheus created
configmap/prometheus created
clusterrole.rbac.authorization.k8s.io/prometheus created
clusterrolebinding.rbac.authorization.k8s.io/prometheus created
service/prometheus created
deployment.apps/prometheus created
```

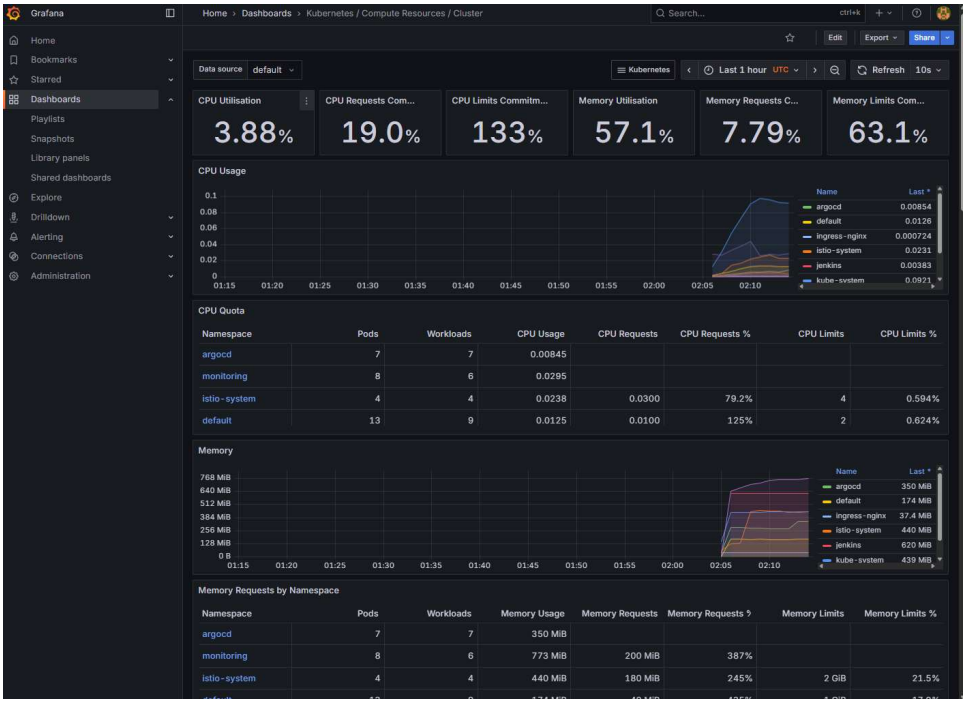
6. **Access Grafana Dashboard**

   Port-forward grafana

```
kubectl -n monitoring port-forward svc/prometheus-grafana 3000:80 &
```

   Go to : http://localhost:3000





7. **Install Kiali**

Kiali is an observability console for Istio servicemesh used to visualise the service mesh topology, monitor traffic, analyse metrics, view distributed traces and validate Istio configs.

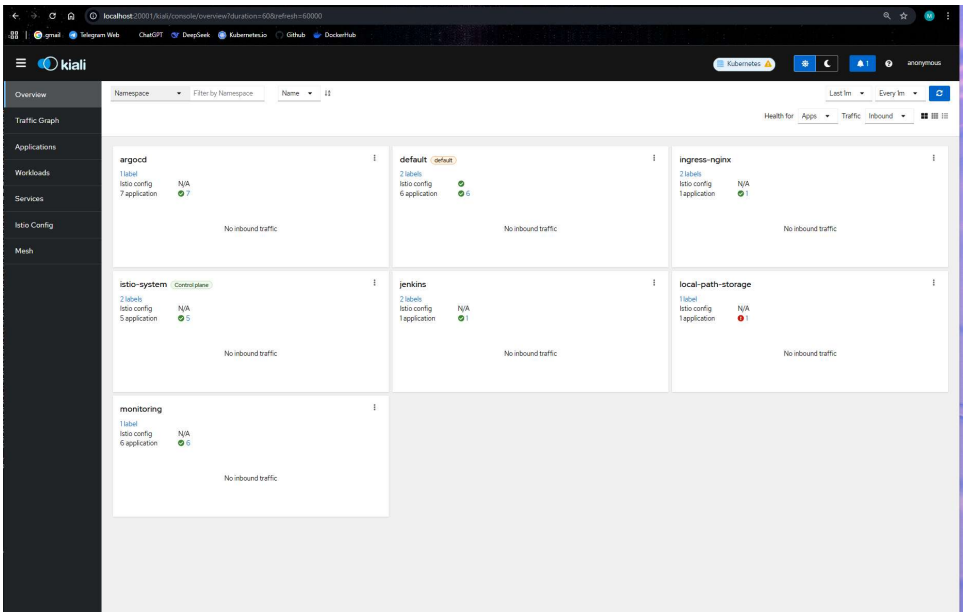
Helps us understand app health, identify bottlenecks and troubleshoot issues within the complex architecture of a microservices environment.

```
# kubectl apply -f
https://raw.githubusercontent.com/istio/istio/release-1.27/samples/addons/kiali.yaml
serviceaccount/kiali created
configmap/kiali created
clusterrole.rbac.authorization.k8s.io/kiali created
clusterrolebinding.rbac.authorization.k8s.io/kiali created
service/kiali created
deployment.apps/kiali created


# k get all -A | grep kiali
istio-system      pod/kiali-56f54f58f9-5xsgj                    1/1     Running     0
103s
istio-system   service/kiali                                ClusterIP   10.43.72.137   <none>
20001/TCP,9090/TCP                                  103s
istio-system      deployment.apps/kiali                         1/1   1       1       103s
istio-system      replicaset.apps/kiali-56f54f58f9                    1       1       1     103s
```

Portforward

```
kubectl port-forward svc/kiali 20001:20001 -n istio-system &
```

On browser, go to http://localhost:20001


localhost:20001/kiali/console/overview?duration=60&refresh=60000
gmail
Telegram Web
ChatGPT
DeepSeek
Kubernetes.io
Github
DockerHub
kiali
Kubernetes
anonymous
Overview
Traffic Graph
Applications
Workloads
Services
Istio Config
Mesh
Namespace
Filter by Namespace
Name
Last 1m
Every 1m
Health for
Apps
Traffic
Inbound
argocd
1 label
Istio config N/A
7 application 7
No inbound traffic
default
default
2 labels
Istio config
6 application 6
No inbound traffic
ingress-nginx
2 labels
Istio config N/A
1 application 1
No inbound traffic
istio-system
Control plane
2 labels
Istio config N/A
5 application 5
No inbound traffic
jenkins
2 labels
Istio config N/A
1 application 1
No inbound traffic
local-path-storage
1 label
Istio config N/A
1 application 1
No inbound traffic
monitoring
1 label
Istio config N/A
6 application 6
No inbound traffic

Troubleshooting Log 1

Sunday, 28 September 2025 11:04 pm

SECTION: GITHUB RESYNC

ISSUE 1: ERROR BUILDING IMAGE

```
# docker buildx build -f sampapp/Dockerfiles/sampapp/Dockerfile --load -t mikiangelica/sampapp:v2 .
[+] Building 6.8s (6/9)
docker:desktop-linux
 => [internal] load build definition from Dockerfile
0.0s
 => => transferring dockerfile: 294B
0.0s
 => [internal] load metadata for docker.io/library/python:3.10-slim
0.9s
 => [internal] load .dockerignore
0.0s
 => => transferring context: 2B
0.0s
 => [1/5] FROM docker.io/library/python:3.10-
slim@sha256:f8081b61393cc16b2339d377bb523a5646cdd28fc6105612b3893c405d27f730
5.8s
 => => resolve docker.io/library/python:3.10-
slim@sha256:f8081b61393cc16b2339d377bb523a5646cdd28fc6105612b3893c405d27f730
0.0s
 => => sha256:fceec8b8a90ef53ff8f6211082b139436e3b3f0f16adf7752fe0d9e6681a3231 250B / 250B
0.6s
 => => sha256:6dc114d5e12c107652ba61ef068cfacc611596390cbafa6cd21206a670e4e264 14.09MB /
14.09MB                                                                    4.6s
 => => sha256:d02c566428bb557aab93bf3d9d768b9abb145010f1f8a6756db930f0902f60f7 1.29MB /
1.29MB                                                                     1.6s
 => => sha256:ce1261c6d567efa8e3b457673eeeb474a0a8066df6bb95ca9a6a94a31e219dd3 15.73MB /
29.77MB                                                                    5.6s
 => [internal] load build context
1.1s
 => => transferring context: 45.81MB
1.0s
 => CANCELED [2/5] WORKDIR /app
0.0s
ERROR: failed to build: failed to solve: Canceled: context canceled
```

^^ Above means build context was too big so it was cancelled.

So the fix was to only include the sampapp dir (context build)

```
# docker buildx build \
  -f sampapp/Dockerfiles/sampapp/Dockerfile \
 -t mikiangelica/sampapp:v2 \
 --load \
 sampapp/
```

SECTION: ANSIBLE TO PROVISION A NEW NODE

ISSUE 2: Playbook not running because role not found

Got this error

```
miki@ansible:~/ansible-k8s$ ansible-playbook -i inventory/inventory.ini playbooks/add-worker.yaml
ERROR! the role 'common' was not found in /home/miki/ansible-k8s/playbooks/roles:/home/miki/.ansible/roles:/usr/share/ansible/roles:/etc/ansible/roles:/home/miki/ansible-k8s/playbooks

The error appears to be in '/home/miki/ansible-k8s/playbooks/add-worker.yaml': line 9, column 7, but may
be elsewhere in the file depending on the exact syntax problem.

The offending line appears to be:

  roles:
    - common
      ^ here
```

Moved the roles/ dir to under playbooks/ dir and the error msg changed, meaning now it can detect

SECTION: ANSIBLE TO PROVISION A NEW NODE

ISSUE 3: Playbook not running because passwordless sudo not configured for user

Got this error

```
miki@ansible:~/ansible-k8s$ ansible-playbook -i inventory/inventory.ini playbooks/add-worker.yaml

PLAY [Configure and join new Kubernetes worker node] ***************************************************************************

TASK [Gathering Facts] *********************************************************************************************************
fatal: [worker2]: FAILED! => {"msg": "Missing sudo password"}
fatal: [worker1]: FAILED! => {"msg": "Missing sudo password"}

PLAY RECAP *********************************************************************************************************************
worker1                    : ok=0    changed=0    unreachable=0    failed=1    skipped=0    rescued=0    ignored=0
worker2                    : ok=0    changed=0    unreachable=0    failed=1    skipped=0    rescued=0    ignored=0
```

Went into each host (master and both workers) to add passwordless sudo for user miki

```
# sudo visudo
# add "miki ALL=(ALL) NOPASSWD:ALL" to end of file
Then save and exit (Ctrl+O, Enter, Ctrl+X in nano).
```

Test sudo cmmand

```
ansible k8s_workers -i inventory/inventory.ini -m shell -a "sudo whoami"
```

The above command hung

```
# ssh worker1 (from ansible)

# sudo -n whoami
sudo: a password is required

# sudo -l -U miki
Matching Defaults entries for miki on worker1:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin
\:/bin\:/snap/bin, use_pty

User miki may run the following commands on worker1:
    (ALL) NOPASSWD: ALL
    (ALL : ALL) ALL
```

The sudo -l output proves miki *is* allowed NOPASSWD, but sudo -n whoami still requiring a password means something else on the node is overriding or taking precedence.

Added an explicit sudoers file
This will create a small file that explicitly grants NOPASSWD with a predictable filename so it sorts last

```
# sudo visudo -f /etc/sudoers.d/99-ansible-miki
# Add "miki ALL=(ALL) NOPASSWD:ALL"
```

Test locally on node

```
sudo -n whoami && echo OK || echo FAIL:$?
root
OK
```

Check SSH and sudo from ansible node

```
# ansible all -i inventory/inventory.ini -m command -a "sudo -n whoami"
[WARNING]: Platform linux on host master1 is using the discovered Python interpreter at
/usr/bin/python3.10, but future installation of another Python interpreter
could change the meaning of that path. See https://docs.ansible.com/ansible-
core/2.17/reference_appendices/interpreter_discovery.html for more information.
master1 | CHANGED | rc=0 >>
root
[WARNING]: Platform linux on host worker2 is using the discovered Python interpreter at
/usr/bin/python3.10, but future installation of another Python interpreter
could change the meaning of that path. See https://docs.ansible.com/ansible-
core/2.17/reference_appendices/interpreter_discovery.html for more information.
worker2 | CHANGED | rc=0 >>
root
[WARNING]: Platform linux on host worker1 is using the discovered Python interpreter at
/usr/bin/python3.10, but future installation of another Python interpreter
could change the meaning of that path. See https://docs.ansible.com/ansible-
core/2.17/reference_appendices/interpreter_discovery.html for more information.
worker1 | CHANGED | rc=0 >>
root
```

Now run again

```
# ansible-playbook -i inventory/inventory.ini playbooks/add-worker.yaml
```

SECTION: ISTIO SERVICEMESH

ISSUE 4: GATEWAY NOT BECOMING PROGRAMMED "TRUE"

Below error

```
$ kubectl get gateway sampapp-gateway -n default -o yaml | grep -A10 status:
status:
  conditions:
  - lastTransitionTime: "2025-10-04T20:13:07Z"
    message: Resource accepted
    observedGeneration: 1
    reason: Accepted
    status: "True"
    type: Accepted
  - lastTransitionTime: "2025-10-04T20:13:07Z"
    message: 'Assigned to service(s) sampapp-gateway-istio.default.svc.cluster.local:80,
      but failed to assign to all requested addresses: address pending for hostname
      "sampapp-gateway-istio.default.svc.cluster.local"'
    observedGeneration: 1
    reason: AddressNotAssigned
    status: "False"
    type: Programmed
  listeners:
  - attachedRoutes: 0
    conditions:
    - lastTransitionTime: "2025-10-04T20:13:07Z"
      message: No errors found
      observedGeneration: 1
      reason: Accepted
      status: "True"
      type: Accepted
    - lastTransitionTime: "2025-10-04T20:13:07Z"
      message: No errors found
      observedGeneration: 1
      reason: NoConflicts
      status: "False"
      type: Conflicted
    - lastTransitionTime: "2025-10-04T20:13:07Z"
      message: No errors found
      observedGeneration: 1
      reason: Programmed
      status: "True"
      type: Programmed
    - lastTransitionTime: "2025-10-04T20:13:07Z"
      message: No errors found
      observedGeneration: 1
      reason: ResolvedRefs
      status: "True"
      type: ResolvedRefs
    name: http
    supportedKinds:
    - group: gateway.networking.k8s.io
      kind: HTTPRoute
    - group: gateway.networking.k8s.io
```

```
kind: GRPCRoute
```

In Istio **1.27+**, the new Gateway API **does not attach VirtualServices anymore**.
It only attaches **HTTPRoute** or **GRPCRoute** resources.

Created HTTPRoute instead and deleted the VS

```
apiVersion: gateway.networking.k8s.io/v1
kind: HTTPRoute
metadata:
 name: sampapp
 namespace: default
spec:
 parentRefs:
 - name: sampapp-gateway
 rules:
 - matches:
   - path:
      type: PathPrefix
      value: /
   backendRefs:
   - name: sampapp-service   # must match your Service name
     port: 5000            # must match your Service port
```

What's Happening

☑ GatewayClass istio → **Accepted = True**
☑ Istiod logs show it's actively reconciling your gateway
❌ Gateway status = Programmed=False (AddressNotAssigned)
👉 **This means Istio *did* pick up your Gateway, but it cannot bind it to any Service (address).**

Why That Happens

Since Istio 1.27+ uses the **new Kubernetes Gateway API**, your Gateway spec:

gatewayClassName: istio
tells Istiod to automatically **spawn a service** named:

sampapp-gateway-istio
in the **same namespace** (default).
That service is responsible for binding a NodePort or LoadBalancer IP —
but in your logs:

failed to assign to all requested addresses: address pending for hostname "sampapp-gateway-istio.default.svc.cluster.local"
…it means that service never got created properly.

Checking gateway-istio service is LoadBalancer

```
# k get deploy,svc -n default
NAME                         READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/nginx        1/1     1            1           26d
deployment.apps/postgres     1/1     1            1           26d
deployment.apps/redis        1/1     1            1           26d
```

```
deployment.apps/sampapp              2/2    2         2         26d
deployment.apps/sampapp-gateway-istio   1/1    1         1         4h58m

NAME                          TYPE         CLUSTER-IP      EXTERNAL-IP   PORT(S)                AGE
service/kubernetes            ClusterIP    10.43.0.1       <none>        443/TCP                26d
service/nginx-service         ClusterIP    10.43.104.188   <none>        80/TCP                 26d
service/postgres-service      ClusterIP    10.43.178.155   <none>        5432/TCP               26d
service/redis-service         ClusterIP    10.43.146.71    <none>        6379/TCP               26d
service/sampapp-gateway-istio   LoadBalancer   10.43.34.51    <pending>
15021:32516/TCP,80:32683/TCP   4h58m
service/sampapp-service       ClusterIP    10.43.133.187   <none>        5000/TCP               26d
```

Patch to NodePort

```
# kubectl patch svc sampapp-gateway-istio -n default \
  -p '{"spec": {"type": "NodePort"}}'
```

Still showed up as LoadBalancer

```
# k get svc sampapp-gateway-istio
NAME                    TYPE           CLUSTER-IP     EXTERNAL-IP   PORT(S)                      AGE
sampapp-gateway-istio   LoadBalancer   10.43.34.51    <pending>     15021:32516/TCP,80:32683/TCP
5h1m
```

Istio's **Gateway controller** is the *owner* of that Service (ownerReferences → gateway.networking.k8s.io/v1beta1), so any manual edit (like changing LoadBalancer → NodePort) is automatically reverted.

kubectl delete gateway sampapp-gateway -n default

New GW yaml

```
apiVersion: gateway.networking.k8s.io/v1
kind: Gateway
metadata:
  name: sampapp-gateway
  namespace: default
  annotations:
    networking.istio.io/service-type: "NodePort"  # 👈 tells Istio to use NodePort
spec:
  gatewayClassName: istio
  listeners:
  - name: http
    port: 80
    protocol: HTTP
    allowedRoutes:
      namespaces:
        from: Same
```

Check

```
$ k get svc sampapp-gateway-istio
NAME                    TYPE       CLUSTER-IP     EXTERNAL-IP   PORT(S)                       AGE
sampapp-gateway-istio   NodePort   10.43.252.215  <none>        15021:30857/TCP,80:31747/TCP   22s

$ k get gateway
```

```
NAME            CLASS   ADDRESS                                   PROGRAMMED   AGE
sampapp-gateway   istio   sampapp-gateway-istio.default.svc.cluster.local   True         41s
```

Troubleshooting Log 2

Sunday, 21 September 2025 3:43 pm

*****ALL BELOW ISSUES ARE FROM PVC FOR POSTGRES SECTION.**

4 ISSUES

ISSUE 1: Pod, PVC and PV not coming up.
Ran into an **ISSUE** where after applying, no pv was made. Pod was not coming up and the pvc was pending state with error saying ProvisioningFailed.

1. Checked the provisioner pod status:

```
# kubectl get po -n local-path-storage
No resources found in local-path-storage namespace.
```

!!! Provisioner pod isn't even running -- that's why PVCs are stuck in Pending and failing to provision.
This pod is important as it is the one actually creating the local PVs.
Without that pod, no one is acting on the PVC requests.

1. Redeploy the local-path-provisioner into cluster: (below steps is for rancher/k3s)

Normally it is pre-installed but if missing, can reapply manifest

```
# kubectl apply -f https://raw.githubusercontent.com/rancher/local-path-provisioner/master/deploy/local-path-storage.yaml

namespace/local-path-storage created
serviceaccount/local-path-provisioner-service-account created
role.rbac.authorization.k8s.io/local-path-provisioner-role created
Warning: resource clusterroles/local-path-provisioner-role is missing the kubectl.kubernetes.io/last-applied-configuration annotation which is required by kubectl apply. kubectl apply should only be used on resources created declaratively by either kubectl create --save-config or kubectl apply. The missing annotation will be patched automatically.
clusterrole.rbac.authorization.k8s.io/local-path-provisioner-role configured
rolebinding.rbac.authorization.k8s.io/local-path-provisioner-bind created
Warning: resource clusterrolebindings/local-path-provisioner-bind is missing the kubectl.kubernetes.io/last-applied-configuration annotation which is required by kubectl apply. kubectl apply should only be used on resources created declaratively by either kubectl create --save-config or kubectl apply. The missing annotation will be patched automatically.
clusterrolebinding.rbac.authorization.k8s.io/local-path-provisioner-bind configured
deployment.apps/local-path-provisioner created
Warning: resource storageclasses/local-path is missing the kubectl.kubernetes.io/last-applied-configuration annotation which is required by kubectl apply. kubectl apply should only be used on resources created declaratively by either kubectl create --save-config or kubectl apply. The missing annotation will be patched automatically.
storageclass.storage.k8s.io/local-path configured
configmap/local-path-config created
```

^^Above manifest will:

- Create local-path-storage namespace
- Deploy the local-path-provisioner pod
- Bind it to your SC (local-path)

ISSUE 2: Pod still not coming up with 'calico' failure error (ContainerCreating state) - Calico Pods Issue
Next **ISSUE**, after applying above manifest, pod is just in ContainerCreating state.

```
Events:
  Type     Reason                  Age                From               Message
  ----     ------                  ----               ----               -------
  Normal   Scheduled               9m3s               default-scheduler  Successfully assigned local-path-
storage/local-path-provisioner-7d6dddf9dd-7hhgx to master1
  Warning  FailedCreatePodSandBox  9m3s               kubelet            Failed to create pod sandbox: rpc
error: code = Unknown desc = failed to setup network for sandbox
"e2bd24f5fc422df3c85ce9880d1eda6265de9ea7e95ab2896799bd0d5cbf4d28": plugin type="calico"
failed (add): error getting ClusterInformation: connection is unauthorized: Unauthorized
  Warning  FailedCreatePodSandBox  8m52s              kubelet            Failed to create pod sandbox: rpc
error: code = Unknown desc = failed to setup network for sandbox
"11b27fe7c8fde4061d54627b9531dcb9ad93518a9ead1fb947f71973ac7b4ebd": plugin type="calico"
failed (add): error getting ClusterInformation: connection is unauthorized: Unauthorized
  Warning  FailedCreatePodSandBox  8m41s              kubelet            Failed to create pod sandbox: rpc
error: code = Unknown desc = failed to setup network for sandbox
"108a10b7d50adee59c81156bafc45a4fccd9c23441f6fb3ee4e91d12e7778114": plugin type="calico"
failed (add): error getting ClusterInformation: connection is unauthorized: Unauthorized
  Warning  FailedCreatePodSandBox  8m28s              kubelet            Failed to create pod sandbox: rpc
error: code = Unknown desc = failed to setup network for sandbox
"9755e3a358659242db0a6b1da3577e4d9a2822f49c6ab9404d6ed48e91ad1e97": plugin type="calico"
failed (add): error getting ClusterInformation: connection is unauthorized: Unauthorized
  Warning  FailedCreatePodSandBox  8m13s              kubelet            Failed to create pod sandbox: rpc
error: code = Unknown desc = failed to setup network for sandbox
"9a3481abb8a92206fc2a21f6eab3ccf9f2f091f3738293b3ea89ab184b0b441c": plugin type="calico"
failed (add): error getting ClusterInformation: connection is unauthorized: Unauthorized
  Warning  FailedCreatePodSandBox  8m2s               kubelet            Failed to create pod sandbox: rpc
error: code = Unknown desc = failed to setup network for sandbox
"ac04750ab380279e62b83f8d5a6eefad39ecf1ffeef9b119112f89767e4a43f3": plugin type="calico"
failed (add): error getting ClusterInformation: connection is unauthorized: Unauthorized
  Warning  FailedCreatePodSandBox  7m46s              kubelet            Failed to create pod sandbox: rpc
error: code = Unknown desc = failed to setup network for sandbox
"18b0b1b51e7d2454d74655518e41cac1e119664583731cd466e8470c08fd7d3d": plugin type="calico"
failed (add): error getting ClusterInformation: connection is unauthorized: Unauthorized
  Warning  FailedCreatePodSandBox  7m31s              kubelet            Failed to create pod sandbox: rpc
error: code = Unknown desc = failed to setup network for sandbox
"8fba02df518f3d586b24610d3f0ddc9b425e53b4106552fcadcfed4a110cec1a": plugin type="calico"
failed (add): error getting ClusterInformation: connection is unauthorized: Unauthorized
  Warning  FailedCreatePodSandBox  7m18s              kubelet            Failed to create pod sandbox: rpc
error: code = Unknown desc = failed to setup network for sandbox
"34d789f5e90931248da4026fb8563d9c685cc409e976031bd32eb4fadf6b9d8b": plugin type="calico"
failed (add): error getting ClusterInformation: connection is unauthorized: Unauthorized
  Warning  FailedCreatePodSandBox  3m29s (x17 over 7m5s)  kubelet            (combined from similar
events): Failed to create pod sandbox: rpc error: code = Unknown desc = failed to setup network for
sandbox "aef83303396dff043d10fe805d439f29ceefc3fb0f1a20236f7fa81e99d902e1": plugin
type="calico" failed (add): error getting ClusterInformation: connection is unauthorized: Unauthorized
```

From above, the pod isnt starting because it cant even get a network interface.

1. Checked calico pods -- all were Running

```
kube-system     calico-kube-controllers-7959b6fcd8-wbcvc          1/1   Running   0
6d7h
kube-system     calico-node-lbzg7                                 1/1   Running   0          6d7h
kube-system     calico-node-wb6lq                                 1/1   Running   0          6d7h
```

1. Checked one calico pod logs and no errors were see.

1. Checked cluster status

```
 # kubectl get clusterinformation.crd.projectcalico.org default -oyaml
apiVersion: crd.projectcalico.org/v1
kind: ClusterInformation
metadata:
  annotations:
    projectcalico.org/metadata: '{"creationTimestamp":"2025-09-08T10:54:36Z"}'
  creationTimestamp: "2025-09-08T10:54:36Z"
  generation: 1
  name: default
  resourceVersion: "1915"
  uid: 25f4597b-b245-4a9f-a542-998ca4d8ac61
spec:
  calicoVersion: v3.30.3
  clusterGUID: 6af15618022245369ad9a7f5dc5e4c7c
  clusterType: k8s,bgp,kdd
  datastoreReady: true
```

1. Checked if I can go into netshoot AFTER **force** deleting it

```
# k delete po netshoot --grace-period=0 --force
```

(if can go in, means that new pods can be created with networking right now)

```
# kubectl exec -it netshoot -- sh
~ #
```

BUT for my case, I recreated and it got the same calico error as above log events for netshoot pod.

1. Tried to reapply calico yaml but still same error as above when trying to restart netshoot pod.

1. Checked for CNI binaries on worker node (shld show calico, calico-ipam. If missing > kubelet cant call calico)

```
# ls -l /opt/cni/bin/ | grep calico

-rwxr-xr-x 1 root root 79640052 Sep  8 10:51 calico
-rwxr-xr-x 1 root root 79640052 Sep  8 10:51 calico-ipam
```

1. Check connectivity to API server… (AKA master node IP but can check with following command):

```
# kubectl config view --minify -o jsonpath='{.clusters[0].cluster.server}'
https://192.168.81.128:6443
```

```
# curl -vk https://192.168.81.128:6443/healthz
*   Trying 192.168.81.128:6443...
* Connected to 192.168.81.128 (192.168.81.128) port 6443 (#0)
* ALPN, offering h2
* ALPN, offering http/1.1
* TLSv1.0 (OUT), TLS header, Certificate Status (22):
* TLSv1.3 (OUT), TLS handshake, Client hello (1):
* TLSv1.2 (IN), TLS header, Certificate Status (22):
* TLSv1.3 (IN), TLS handshake, Server hello (2):
* TLSv1.2 (IN), TLS header, Finished (20):
* TLSv1.2 (IN), TLS header, Supplemental data (23):
* TLSv1.3 (IN), TLS handshake, Encrypted Extensions (8):
* TLSv1.2 (IN), TLS header, Supplemental data (23):
* TLSv1.3 (IN), TLS handshake, Request CERT (13):
* TLSv1.2 (IN), TLS header, Supplemental data (23):
* TLSv1.3 (IN), TLS handshake, Certificate (11):
* TLSv1.2 (IN), TLS header, Supplemental data (23):
* TLSv1.3 (IN), TLS handshake, CERT verify (15):
* TLSv1.2 (IN), TLS header, Supplemental data (23):
* TLSv1.3 (IN), TLS handshake, Finished (20):
* TLSv1.2 (OUT), TLS header, Finished (20):
* TLSv1.3 (OUT), TLS change cipher, Change cipher spec (1):
* TLSv1.2 (OUT), TLS header, Supplemental data (23):
* TLSv1.3 (OUT), TLS handshake, Certificate (11):
* TLSv1.2 (OUT), TLS header, Supplemental data (23):
* TLSv1.3 (OUT), TLS handshake, Finished (20):
* SSL connection using TLSv1.3 / TLS_AES_128_GCM_SHA256
* ALPN, server accepted to use h2
* Server certificate:
*  subject: O=k3s; CN=k3s
*  start date: Sep  8 10:09:29 2025 GMT
*  expire date: Sep  8 10:09:29 2026 GMT
*  issuer: CN=k3s-server-ca@1757326169
*  SSL certificate verify result: self-signed certificate in certificate chain (19), continuing anyway.
* Using HTTP2, server supports multiplexing
* Connection state changed (HTTP/2 confirmed)
* Copying HTTP/2 data in stream buffer to connection buffer after upgrade: len=0
* TLSv1.2 (OUT), TLS header, Supplemental data (23):
* TLSv1.2 (OUT), TLS header, Supplemental data (23):
* TLSv1.2 (OUT), TLS header, Supplemental data (23):
* Using Stream ID: 1 (easy handle 0x555a88c279f0)
* TLSv1.2 (OUT), TLS header, Supplemental data (23):
> GET /healthz HTTP/2
> Host: 192.168.81.128:6443
> user-agent: curl/7.81.0
> accept: */*
>
* TLSv1.2 (IN), TLS header, Supplemental data (23):
* TLSv1.3 (IN), TLS handshake, Newsession Ticket (4):
* TLSv1.2 (IN), TLS header, Supplemental data (23):
* Connection state changed (MAX_CONCURRENT_STREAMS == 250)!
* TLSv1.2 (OUT), TLS header, Supplemental data (23):
* TLSv1.2 (IN), TLS header, Supplemental data (23):
```

```
* TLSv1.2 (IN), TLS header, Supplemental data (23):
* TLSv1.2 (IN), TLS header, Supplemental data (23):
* TLSv1.2 (IN), TLS header, Supplemental data (23):
< HTTP/2 401
< audit-id: 1e53d3b7-f894-4988-8cf3-4a7656ae33af
< cache-control: no-cache, private
< content-type: application/json
< content-length: 157
< date: Mon, 15 Sep 2025 01:22:59 GMT
<
* TLSv1.2 (IN), TLS header, Supplemental data (23):
{
  "kind": "Status",
  "apiVersion": "v1",
  "metadata": {},
  "status": "Failure",
  "message": "Unauthorized",
  "reason": "Unauthorized",
  "code": 401
* Closing connection 0
* TLSv1.2 (OUT), TLS header, Supplemental data (23):
* TLSv1.3 (OUT), TLS alert, close notify (256):
}root@worker1:~# kubectl config view --minify -ojsonpath='{.clusters[0].cluster.server}'
error: current-context must exist in order to minify
root@worker1:~#
logout
miki@worker1:~$
logout
Connection to 192.168.81.129 closed.
```

^^The output above shows that we **successfully** connected to the API server. TLS handshake succeeded and server responded so there's no firewall or routing issue.
The response says :
"
  "status": "Failure",
  "message": "Unauthorized",
  "reason": "Unauthorized",
  "code": 401"
Which is normal, because curl didn't send a client cert/token. (curl isn't authenticated)

1. Check that the CNI config file exists

```
root@worker1:~# cat /etc/cni/net.d/10-calico.conflist

{
  "name": "k8s-pod-network",
  "cniVersion": "0.3.1",
  "plugins": [
    {
      "type": "calico",
      "log_level": "info",
      "log_file_path": "/var/log/calico/cni/cni.log",
      "datastore_type": "kubernetes",
      "nodename": "worker1",
      "mtu": 0,
```

```
    "ipam": {
        "type": "calico-ipam"
    },
    "policy": {
        "type": "k8s"
    },
    "kubernetes": {
        "kubeconfig": "/etc/cni/net.d/calico-kubeconfig"
    }
   },
   {
    "type": "portmap",
    "snat": true,
    "capabilities": {"portMappings": true}
   }
 ]
```

^^Shld be a valid JSON file referencing calico as type.
If missing or corrupt > kubelet cant set up networking

1. Checked if calico crds are present. If these aren't applied correctly, calico won't run. Below shows that the crds are healthy and loaded.

```
# kubectl get crd | grep projectcalico
bgpconfigurations.crd.projectcalico.org             2025-09-15T02:29:01Z
bgpfilters.crd.projectcalico.org                    2025-09-08T10:48:34Z
bgppeers.crd.projectcalico.org                      2025-09-15T02:29:01Z
blockaffinities.crd.projectcalico.org               2025-09-15T02:29:01Z
caliconodestatuses.crd.projectcalico.org            2025-09-15T02:29:01Z
clusterinformations.crd.projectcalico.org           2025-09-15T02:29:01Z
felixconfigurations.crd.projectcalico.org           2025-09-15T02:29:01Z
globalnetworkpolicies.crd.projectcalico.org         2025-09-15T02:29:01Z
globalnetworksets.crd.projectcalico.org             2025-09-15T02:29:01Z
hostendpoints.crd.projectcalico.org                 2025-09-15T02:29:01Z
ipamblocks.crd.projectcalico.org                    2025-09-15T02:29:01Z
ipamconfigs.crd.projectcalico.org                   2025-09-15T02:29:01Z
ipamhandles.crd.projectcalico.org                   2025-09-15T02:29:01Z
ippools.crd.projectcalico.org                       2025-09-15T02:29:01Z
ipreservations.crd.projectcalico.org                2025-09-15T02:29:01Z
kubecontrollersconfigurations.crd.projectcalico.org 2025-09-15T02:29:01Z
networkpolicies.crd.projectcalico.org               2025-09-15T02:29:01Z
networksets.crd.projectcalico.org                   2025-09-15T02:29:01Z
stagedglobalnetworkpolicies.crd.projectcalico.org   2025-09-08T10:48:34Z
stagedkubernetesnetworkpolicies.crd.projectcalico.org 2025-09-08T10:48:34Z
stagednetworkpolicies.crd.projectcalico.org          2025-09-08T10:48:34Z
tiers.crd.projectcalico.org                         2025-09-08T10:48:34Z
```

1. Checked the kubeconfig file in worker node. In the output from #8, we cat the file that is stated under kubernetes, kubeconfig

```
# cat /etc/cni/net.d/calico-kubeconfig
# Kubeconfig file for Calico CNI plugin. Installed by calico/node.
apiVersion: v1
kind: Config
```

```
clusters:
- name: local
  cluster:
    server: https://10.43.0.1:443
    certificate-authority-data:
"LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUJkekNDQVIyZ0F3SUJBZ0lCQURBS0JnZ3Foa2pPUFFRR
EFqQWpNU0V3SHdZRFZRUUREQmhyTTNNdGMyVnkKZG1WeUxXTmhRREUzTlRjek1qWXhOamt3SGhjTk
1qVXdPVEE0TVRBd09USTVXaGNOTXpVd09UQTJNVEF3T1RJNQpXakFqTVNFd0h3WURWUVFEREJock0zT
XRjMlZ5ZG1WeUxXTmhRREUzTlRjek1qWXhOamt3V1RBVEJnY3Foa2pPClBRSUJCZ2dxaGtqT1BRTUJCd05
DQUFRSHNMZmF4alN5T3U0a1VacVlNK211UGUwUzdGRTd1ZEErM05jNmZLaDIKQytibTRqZ2RKUVYrV2l
yWkFZc3MwZllZdEFxN05PVHNqUzBDYXJMeDcxMXlvMEl3UURBT0JnTlZIUThCQWY4RQpCQU1DQXFRd0
R3WURWUjBUQVFIL0JBVXdBd0VCL3pBZEJnTlZIUTRFRmdRVTI0b1R1cVVtckRybWNFcXQrdm5GCk5NNm
EzWVF3Q2dZSUtvWkl6ajBFQXdJRFNBQXdSUUlnZXpIVGFNeFFPUDQyQ0dCOUFLUGtQVDluWXVaZy8wRl
QKQ2MzeXlQdjJFVUVDSVFEQmdZWjVjRTVucEZvd3h3QkNhOEhZaENxK0NKc3A4TGFuN3lUZzVrVWFlQT
09Ci0tLS0tRU5EIENFUlRJRklDQVRFLS0tLS0K"
users:
- name: calico
  user:
    token:
eyJhbGciOiJSUzI1NiIsImtpZCI6IjJDTV9rZzNneFc5RkVIdWpuUTh2cktHVC1acmFlRDlfalRscFBqUUJ5Qmcif
Q.eyJhdWQiOlsiaHR0cHM6Ly9rdWJlcm5ldGVzLmRlZmF1bHQuc3ZjLmNsdXN0ZXIubG9jYWwiLCJrM3MiX
SwiZXhwIjoxNzU3OTg1MzQ0LCJpYXQiOjE3NTc4OTg5NDQsImlzcyI6Imh0dHBzOi8va3ViZXJuZXRlcy5kZW
ZhdWx0LnN2Yy5jbHVzdGVyLmxvY2FsIiwianRpIjoiYTQ1OGU0NTMtZTk1Yi00ZTkyLWEyY2ItMGI2Y2Y0Nzk
5MzQ2Iiwia3ViZXJuZXRlcy5pbyI6eyJuYW1lc3BhY2UiOiJrdWJlLXN5c3RlbSIsInNlcnZpY2VhY2NvdW50Ijp7I
m5hbWUiOiJjYWxpY28tY25pLXBsdWdpbiIsInVpZCI6IjA2NDMyNTIzLWQyN2MtNDg2ZS05ZmNkLTkxY2Q
wMWQ1MTc2OCJ9fSwibmJmIjoxNzU3ODk4OTQ0LCJzdWIiOiJzeXN0ZW06c2VydmljZWFjY291bnQ6a3Vi
ZS1zeXN0ZW06Y2FsaWNvLWNuaS1wbHVnaW4ifQ.MKmSIQ1dvVF7Rt93RA-
PjGQxeptrjrsjUm7ceC4rXO2OCYh1av3TNdbsFcxMSGok41H7z0n3j4jnnowi8l3YXRz0C26XkUk2MPsTgzVP
glj3xS2Fhl5oZ_TgSk3g3_mRUggS_SrKh_UgUd7JImHnSPG6_BYpAx4wQNvrEH-
BX3fIcI6vYymrsURabETLmhnusntrqCYhS7
_M2DmwNztswaL7D45W_ho2EwdhS2NDDZ7Ez8mXqZCQ2TDmV3SGqf3Qn23bL8v_B1gLjSTRJmcCS4Ub
K5BcJHOYrb7MXWY6asHx3GQzIZLw5gqarskMUvCWCz1T1jOqcu62MqZGlbrN5Q
contexts:
- name: calico-context
  context:
    cluster: local
    user: calico
```

^^We see in the cluster server section, the IP stated does not match out master node IP.

The IP stated is the Kubernetes service ClusterIP for kubernetes.default.svc. Normally this is fine bc calico CNI shld be able to reach the API server via the cluster DNS and kube-proxy. But in this case, either kube-proxy isn't working properly or the service IP isn't reachable on the worker node which breaks calico CNI API calls.

11. I also did a curl to the ClusterIP API:

```
curl -k https://10.43.0.1:443/healthz
{
  "kind": "Status",
  "apiVersion": "v1",
  "metadata": {},
  "status": "Failure",
  "message": "Unauthorized",
  "reason": "Unauthorized",
```

```
"code": 401
```

^^From above we can see that there is a 401 unauthorized error **not a timeout** -

- so the service CIDR **is reachable from worker node**.
- Kube-proxy is routing traffic correctly
- The 401 Unauthorized is just bc curl isn't passing credentials
- So networking to the API svc is fine. That means calico **shld be able to talk to the API via 10.43.0.1**

1. I reapplied calico manifest again and checked pods, they are not coming up now

```
# kubectl get po -A | grep -i calico
kube-system     calico-kube-controllers-7498b9bb4c-9cfq9          0/1   Terminating   0
10m
kube-system     calico-node-gprjf                                 0/1   Running       0          6m28s
kube-system     calico-node-s4txj                                 0/1   Init:2/3      0          4m7s
```

1. Checked logs of one of the calico pod

```
  Warning  Unhealthy  3m55s (x2 over 3m57s)  kubelet          Readiness probe failed: calico/node is not
ready: BIRD is not ready: Error querying BIRD: unable to connect to BIRDv4 socket: dial unix
/var/run/calico/bird.ctl: connect: connection refused
  Warning  Unhealthy  3m46s                  kubelet          Readiness probe failed: 2025-09-15 01:52:15.865
[INFO][209] node/health.go 202: Number of node(s) with BGP peering established = 0
calico/node is not ready: BIRD is not ready: BGP not established with 192.168.81.129
  Warning  Unhealthy  3m36s  kubelet  Readiness probe failed: 2025-09-15 01:52:25.876 [INFO][245]
node/health.go 202: Number of node(s) with BGP peering established = 0
calico/node is not ready: BIRD is not ready: BGP not established with 192.168.81.129
  Warning  Unhealthy  3m26s  kubelet  Readiness probe failed: 2025-09-15 01:52:35.866 [INFO][280]
node/health.go 202: Number of node(s) with BGP peering established = 0
calico/node is not ready: BIRD is not ready: BGP not established with 192.168.81.129
  Warning  Unhealthy  3m16s  kubelet  Readiness probe failed: 2025-09-15 01:52:45.872 [INFO][307]
node/health.go 202: Number of node(s) with BGP peering established = 0
calico/node is not ready: BIRD is not ready: BGP not established with 192.168.81.129
  Warning  Unhealthy  3m6s  kubelet  Readiness probe failed: 2025-09-15 01:52:55.873 [INFO][341]
node/health.go 202: Number of node(s) with BGP peering established = 0
calico/node is not ready: BIRD is not ready: BGP not established with 192.168.81.129
  Warning  Unhealthy  2m56s  kubelet  Readiness probe failed: 2025-09-15 01:53:05.863 [INFO][367]
node/health.go 202: Number of node(s) with BGP peering established = 0
calico/node is not ready: BIRD is not ready: BGP not established with 192.168.81.129
  Warning  Unhealthy  2m46s  kubelet  Readiness probe failed: 2025-09-15 01:53:15.869 [INFO][393]
node/health.go 202: Number of node(s) with BGP peering established = 0
calico/node is not ready: BIRD is not ready: BGP not established with 192.168.81.129
  Warning  Unhealthy  2m36s  kubelet  Readiness probe failed: 2025-09-15 01:53:25.869 [INFO][420]
node/health.go 202: Number of node(s) with BGP peering established = 0
calico/node is not ready: BIRD is not ready: BGP not established with 192.168.81.129
  Warning  Unhealthy  17s (x15 over 2m34s)  kubelet  (combined from similar events): Readiness probe
failed: 2025-09-15 01:55:44.668 [INFO][855] node/health.go 202: Number of node(s) with BGP peering
established = 0
calico/node is not ready: BIRD is not ready: BGP not established with 192.168.81.129
```

^^ From above we can see that it says BGP (BIRD) is not ready…
Calico pods are stuck Not Ready bc BGP (BIRD) cannot establish any peers.
In a 2 node cluster, Calico defaults to using BGP peering between nodes but BGP failing > calico readiness probe fails > CNI fails > New pods (workloads and even calico itself) cant get networking

For small clusters, BGP is not really needed so we can switch calico to run in VXLAN mode (overlay

network). BGP is mainly for bigger, prod clusters where you want explicit routing.

1. Change to VXLAN mode

Edit the default IPPool

```
# kubectl edit ippool default-ipv4-ippool -n kube-system
```

Change

```
vxlanMode: Never
```

To:

```
vxlanMode: Always
```

And natOutgoing shld be true.

Then restart calico pods

```
# kubectl delete po -n kube-system -l k8s-app=calico-node
```

1. I had restarted my VMs (cuz I continued troubleshooting next dday) and now all calico pods were up but when I described 1 calico-node pod it shot out this error

```
Events:
  Type     Reason                  Age    From               Message
  ----     ------                  ----   ----               -------
  Normal   Scheduled               11h    default-scheduler  Successfully assigned kube-system/calico-node-4xrnn to worker1
  Warning  FailedCreatePodSandBox  11h    kubelet            Failed to create pod sandbox: rpc error: code = Unknown desc = failed to start sandbox "f955d7cce9bfcea0d9139017e2c89956c841454e841416e1172ee7217b845f07": failed to get sandbox image "rancher/mirrored-pause:3.6": failed to pull image "rancher/mirrored-pause:3.6": failed to pull and unpack image "docker.io/rancher/mirrored-pause:3.6": failed to resolve reference "docker.io/rancher/mirrored-pause:3.6": failed to authorize: failed to fetch anonymous token: Get "https://auth.docker.io/token?scope=repository%3Arancher%2Fmirrored-pause%3Apull&service=registry.docker.io": net/http: TLS handshake timeout
  Warning  FailedCreatePodSandBox  11h    kubelet            Failed to create pod sandbox: rpc error: code = Unknown desc = failed to start sandbox "667043e5c17804cdb892fa3a90f164c25db34a7097fe7d077bcbaa25876843d2": failed to get sandbox image "rancher/mirrored-pause:3.6": failed to pull image "rancher/mirrored-pause:3.6": failed to pull and unpack image "docker.io/rancher/mirrored-pause:3.6": failed to copy: httpReadSeeker: failed open: failed to authorize: failed to fetch anonymous token: Get "https://auth.docker.io/token?scope=repository%3Arancher%2Fmirrored-pause%3Apull&service=registry.docker.io": net/http: TLS handshake timeout
  Warning  FailedCreatePodSandBox  11h    kubelet            Failed to create pod sandbox: rpc error: code = Unknown desc = failed to start sandbox "b86eef5503797237463e24bfb5d6ab1453cf0ee5973be957516fb63649ee4015": failed to get sandbox image "rancher/mirrored-pause:3.6": failed to pull image "rancher/mirrored-pause:3.6": failed to pull and unpack image "docker.io/rancher/mirrored-pause:3.6": failed to copy: httpReadSeeker: failed open: failed to authorize: failed to fetch anonymous token: Get "https://auth.docker.io/token?scope=repository%3Arancher%2Fmirrored-pause%3Apull&service=registry.docker.io": dial tcp 3.88.79.16:443: i/o timeout
```

```
  Warning  FailedCreatePodSandBox  11h    kubelet          Failed to create pod sandbox: rpc error: code
= Unknown desc = failed to start sandbox
"18fa76bf5588026acdb2648a3ded632beaa516c39bbc455d35df5fbe83ee541a": failed to get sandbox
image "rancher/mirrored-pause:3.6": failed to pull image "rancher/mirrored-pause:3.6": failed to pull
and unpack image "docker.io/rancher/mirrored-pause:3.6": failed to resolve reference
"docker.io/rancher/mirrored-pause:3.6": failed to do request: Head
"https://registry-1.docker.io/v2/rancher/mirrored-pause/manifests/3.6": dial tcp: lookup
registry-1.docker.io: Try again
  Warning  FailedCreatePodSandBox  11h    kubelet          Failed to create pod sandbox: rpc error: code
= DeadlineExceeded desc = failed to start sandbox
"329a34c2df6aac55635978b524bed7d6bb37f4c4bb0fedc33c2e6b4eb6aaeb05": failed to get sandbox
image "rancher/mirrored-pause:3.6": failed to pull image "rancher/mirrored-pause:3.6": failed to pull
and unpack image "docker.io/rancher/mirrored-pause:3.6": failed to resolve reference
"docker.io/rancher/mirrored-pause:3.6": failed to do request: Head
"https://registry-1.docker.io/v2/rancher/mirrored-pause/manifests/3.6": dial tcp 18.213.217.113:443:
i/o timeout
  Normal   SandboxChanged        6m31s  kubelet          Pod sandbox changed, it will be killed and re-
created.
  Normal   Pulling               6m23s  kubelet          Pulling image "docker.io/calico/cni:v3.25.0"
  Normal   Pulled                5m55s  kubelet          Successfully pulled image
"docker.io/calico/cni:v3.25.0" in 30.719s (30.719s including waiting). Image size: 87984941 bytes.
  Normal   Created               5m55s  kubelet          Created container: upgrade-ipam
  Normal   Started               5m54s  kubelet          Started container upgrade-ipam
  Normal   Pulled                5m54s  kubelet          Container image "docker.io/calico/cni:v3.25.0" already
present on machine
  Normal   Created               5m54s  kubelet          Created container: install-cni
  Normal   Started               5m54s  kubelet          Started container install-cni
  Normal   Pulling               5m51s  kubelet          Pulling image "docker.io/calico/node:v3.25.0"
  Normal   Pulled                4m45s  kubelet          Successfully pulled image
"docker.io/calico/node:v3.25.0" in 1m5.376s (1m5.376s including waiting). Image size: 87185935 bytes.
  Normal   Created               4m45s  kubelet          Created container: mount-bpffs
  Normal   Started               4m45s  kubelet          Started container mount-bpffs
  Normal   Pulled                4m45s  kubelet          Container image "docker.io/calico/node:v3.25.0"
already present on machine
  Normal   Created               4m45s  kubelet          Created container: calico-node
  Normal   Started               4m44s  kubelet          Started container calico-node
  Warning  Unhealthy             4m43s  kubelet          Readiness probe failed: calico/node is not ready:
felix is not ready: Get "http://localhost:9099/readiness": dial tcp 127.0.0.1:9099: connect:
connection refused
  Warning  Unhealthy             4m42s  kubelet          Readiness probe failed: calico/node is not ready:
felix is not ready: readiness probe reporting 503
```

Felix is the main dataplane agent inside each calico-node pod. Readiness probe is failing because felix can't start its internal API on port 9099.
Common causes - IPPool misconfiguration(wrong mode like CrossSubnet > BGP dependency)

```
# kubectl get ippool default-ipv4-ippool -oyaml
apiVersion: crd.projectcalico.org/v1
kind: IPPool
metadata:
  annotations:
    projectcalico.org/metadata: '{"uid":"0dbcf429-e106-4774-
bc49-3f615231c793","creationTimestamp":"2025-09-15T13:35:22Z"}'
  creationTimestamp: "2025-09-15T13:35:22Z"
```

```
  generation: 1
  name: default-ipv4-ippool
  resourceVersion: "15462"
  uid: aacb61b2-15e3-4675-93a5-6de217b6520f
spec:
  allowedUses:
  - Workload
  - Tunnel
  blockSize: 26
  cidr: 172.16.0.0/16
  ipipMode: Never
  natOutgoing: true
  nodeSelector: all()
  vxlanMode: CrossSubnet
```

1. Checked that kernel vxlan support is running in both nodes with lsmod | grep vxlan (shld show up) -- if doesn't show up need to use modprobe to load module.
2. Patched ippool vxlanMode to Always

```
kubectl patch ippool default-ipv4-ippool \
> --type merge -p '{"spec": {"vxlanMode": "Always", "ipipMode": "Never", "natOutgoing": true}}'
```

Restarted pods and now this error:

```
Events:
  Type     Reason     Age   From               Message
  ----     ------     ----  ----               -------
  Normal   Scheduled  37s   default-scheduler  Successfully assigned kube-system/calico-node-cd7ld to worker1
  Normal   Pulled     31s   kubelet            Container image "docker.io/calico/cni:v3.25.0" already present on machine
  Normal   Created    31s   kubelet            Created container: upgrade-ipam
  Normal   Started    31s   kubelet            Started container upgrade-ipam
  Normal   Pulled     30s   kubelet            Container image "docker.io/calico/cni:v3.25.0" already present on machine
  Normal   Created    30s   kubelet            Created container: install-cni
  Normal   Started    30s   kubelet            Started container install-cni
  Normal   Pulled     27s   kubelet            Container image "docker.io/calico/node:v3.25.0" already present on machine
  Normal   Created    27s   kubelet            Created container: mount-bpffs
  Normal   Started    27s   kubelet            Started container mount-bpffs
  Normal   Pulled     26s   kubelet            Container image "docker.io/calico/node:v3.25.0" already present on machine
  Normal   Created    26s   kubelet            Created container: calico-node
  Normal   Started    26s   kubelet            Started container calico-node
  Warning  Unhealthy  25s   kubelet            Readiness probe failed: calico/node is not ready: felix is not ready: Get "http://localhost:9099/readiness": dial tcp 127.0.0.1:9099: connect: connection refused
```

Calico pod is starting but readiness is still failing.. The problem is now inside felix itself.

But pods themselves are Running

I created 2 test pods and did a ping test to each other to check pod connectivity

```
kubectl run test1 --image=busybox:1.28 --restart=Never -- sleep 3600
kubectl run test2 --image=busybox:1.28 --restart=Never -- sleep 3600
Kubectl exec -it test1 -- ping <test2 IP>
```

OK this didn't work - I realise because of my netpol default deny all. Ping works in other ns.

1. Suspect maybe due to version mismatch (first time I downloaded calico was 3.30.3 version. After that when I installed it was 3.25x version. So the first crds may have been 3.30.3 and the older version could not work with these crds.)

```
# kubectl delete -f https://raw.githubusercontent.com/projectcalico/calico/v3.30.3/manifests/calico.yaml
# kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.30.3/manifests/calico-vxlan.yaml
```

Now pods are running and checking the felix container itself with

```
# kubectl logs <pod-name> -n kube-system -c calico-node | tail -n 50
```

Checked Readiness of pods directly

```
kubectl -n kube-system get po calico-node-9dgn8 \
> -o jsonpath='{.status.conditions[?(@.type=="Ready")].status}{"\n"}'
True
```

ISSUE 3: CoreDNS failure found

**Due to Config Map misconfig

I saw that I couldn't ping by name when doing the pod to pod connectivity test in nettest ns. But I could ping by pod IP.

1. Checked coreDNS pod is running

```
# kubectl get po -n kube-system -l k8s-app=kube-dns -owide

NAME                       READY   STATUS    RESTARTS      AGE    IP              NODE      NOMINATED NODE   READINESS GATES
coredns-64fd4b4794-csng7   1/1     Running   2 (66m ago)   7d4h   172.16.235.130   worker1   <none>           <none>
```

^^Pod is running

1. Checked svc is running

```
# kubectl get svc kube-dns -n kube-system
NAME       TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)                  AGE
kube-dns   ClusterIP   10.43.0.10    <none>        53/UDP,53/TCP,9153/TCP   7d4h
```

^^Svc exists at 10.43.0.10:53/UDP

1. Checked DNS from pod

```
# k exec -it netshoot -- nslookup kubernetes.default.svc.cluster.local
;; communications error to 10.43.0.10#53: timed out
```

```
;; communications error to 10.43.0.10#53: timed out
;; communications error to 10.43.0.10#53: timed out
;; no servers could be reached

command terminated with exit code 1
```

^^Pods cannot reach the kube-dns Service IP > Service routing or kube-proxy/iptables issue.

1. Tested dns resolution from a dnsutils pod

```
# kubectl run -it dnsutils --image=infoblox/dnstools --restart=Never  --rm --command -- sh
If you don't see a command prompt, try pressing enter.
dnstools# nslookup kubernetes.default.svc.cluster.local
Server:         10.43.0.10
Address:        10.43.0.10#53

** server can't find kubernetes.default.svc.cluster.local.localdomain: SERVFAIL

dnstools# kubectl -n kube-^C
dnstools# ^C
dnstools# E0915 22:53:42.978134    4392 v2.go:104] "Unhandled Error" err="write on closed stream 0"
pod "dnsutils" deleted
pod default/dnsutils terminated (Error)
```

^^ CoreDNS svc is reachable so service/clusterIP routing is fine. But coreDNS fails to resolve queries (SERVFAIL)

1. Checked logs of coredns pod

```
# kubectl -n kube-system logs -l k8s-app=kube-dns
[WARNING] No files matching import glob pattern: /etc/coredns/custom/*.override
[WARNING] No files matching import glob pattern: /etc/coredns/custom/*.server
[WARNING] No files matching import glob pattern: /etc/coredns/custom/*.override
[WARNING] No files matching import glob pattern: /etc/coredns/custom/*.server
[ERROR] plugin/errors: 2 kubernetes.default.svc.cluster.local.localdomain. A: read udp
172.16.235.130:60498->192.168.81.2:53: i/o timeout
[ERROR] plugin/errors: 2 kubernetes.default.svc.cluster.local.localdomain. A: read udp
172.16.235.130:58302->192.168.81.2:53: i/o timeout
[WARNING] No files matching import glob pattern: /etc/coredns/custom/*.override
[WARNING] No files matching import glob pattern: /etc/coredns/custom/*.server
[WARNING] No files matching import glob pattern: /etc/coredns/custom/*.override
[WARNING] No files matching import glob pattern: /etc/coredns/custom/*.server
```

^^ Above shows that coreDNS is trying to resolve kubernetes.default.svc.cluster.local

- Instead of answering from the in-cluster K8s plugin, it's forwarding queries upstream (to 192.168.81.2:53)
- That upstream (prolly host VM's DNS) is not reachable from inside the CoreDNS pod
- So DNS lookup fails with SERVFAIL

1. Amend configmap config

kubectl edit configmap -n kube-system coredns

```
apiVersion: v1
data:
  Corefile: |
    .:53 {
        errors
```

```
    health
    ready
    kubernetes cluster.local in-addr.arpa ip6.arpa {
      pods insecure
      fallthrough in-addr.arpa ip6.arpa
    }
    hosts /etc/coredns/NodeHosts {
      ttl 60
      reload 15s
      fallthrough
    }
    prometheus :9153
    # OLD : forward . /etc/resolv.conf
    # NEW : forward . 8.8.8.8 1.1.1.1
    cache 30
    loop
    reload
    loadbalance
    import /etc/coredns/custom/*.override
  }
  import /etc/coredns/custom/*.server
 NodeHosts: |
  192.168.81.128 master1
  192.168.81.129 worker1
kind: ConfigMap
metadata:
 annotations:
  objectset.rio.cattle.io/applied:
H4sIAAAAAAAA/4yQwWrzMBCEX0Xs2fEf20nsX9BDybH02lMva2kdq1Z2g6SkBJN3L8IUCiVtbyNGOzvfzoA
n90IhOmHQcKmgAIsJQc+wl0CD8wQaSr1t1PzKSilFIUilix4JfRoXHQjtdZHTuafAlCgq488xUSi9wK2AybEFDX
vhwR2e8QQFHCnh50ZkloTJCcf8lP6NTIqUyuCkNJiSp9LJP5czoLjryztTWB0uE2iYmvjFuVSFenJsHx6tFf41gv
GY6Y0Eshz/9D2e0OSZfIJVvMZExwzusSf/I9SIcQQNvaG6a+r/XVdV7abBddPtsN9W66Eedi0N7aberM22za
Hf6t0tcPsIAAD//8Ix+PfoAQAA
  objectset.rio.cattle.io/id: ""
  objectset.rio.cattle.io/owner-gvk: k3s.cattle.io/v1, Kind=Addon
  objectset.rio.cattle.io/owner-name: coredns
  objectset.rio.cattle.io/owner-namespace: kube-system
 creationTimestamp: "2025-09-08T10:09:39Z"
 labels:
  objectset.rio.cattle.io/hash: bce283298811743a0386ab510f2f67ef74240c57
 name: coredns
 namespace: kube-system
 resourceVersion: "559"
 uid: 8f9e5d04-3d23-49a6-8307-6e9e05e7efd6
```

1. Restart deployment

```
# k rollout restart deploy -n kube-system coredns
```

1. Check again

```
# kubectl run -it dnsutils --image=infoblox/dnstools --restart=Never  --rm --command -- sh
If you don't see a command prompt, try pressing enter.
dnstools# nslookup kubernetes.default.svc.cluster.local
```

```
Server:         10.43.0.10
Address:        10.43.0.10#53

Name:   kubernetes.default.svc.cluster.local
Address: 10.43.0.1
```

ISSUE 4: (RELATED TO ISSUE 1) Local-path-storage pods not coming up
Broken config. Need to reapply manifests and also create secret.

1. Upon checking local-path-storage pods, all were not up

```
# k get all -n local-path-storage
NAME                                     READY   STATUS             RESTARTS       AGE
pod/local-path-provisioner-5b5f8bd697-44ss2   0/1     CrashLoopBackOff   3 (20s ago)     65s
pod/local-path-provisioner-7d6dddf9dd-gdbkq   0/1     CrashLoopBackOff   10 (2m7s ago)   28m

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/local-path-provisioner   0/1     1            0           21h

NAME                                           DESIRED   CURRENT   READY   AGE
replicaset.apps/local-path-provisioner-5b5f8bd697   1         1         0       65s
replicaset.apps/local-path-provisioner-7d6dddf9dd   1         1         0       21h
```

1. Logs:

```
# k logs -n local-path-storage pod/local-path-provisioner-5b5f8bd697-44ss2
time="2025-09-15T15:07:38Z" level=fatal msg="Error starting daemon: invalid empty flag helper-pod-file and it also does not exist at ConfigMap local-path-storage/local-path-config with err: configmaps \"local-path-config\" is forbidden: User \"system:serviceaccount:local-path-storage:local-path-provisioner-service-account\" cannot get resource \"configmaps\" in API group \"\" in the namespace \"local-path-storage\""
```

Seems cm local-path-config is missing in the ns. The manifest applied may have been incomplete/mismatched.

1. Force delete broken ns

```
# kubectl delete ns local-path-storage --force --grace-period=0
```

Terminating state for so long because of the finalizer (kubernetes)

When doing k get all -n local-path-storage, it showed no resources but ns was stuck in terminating state. This means that the ns is already empty but only the ns object itself is stuck in Term state bc of its own finalizer.

1. Save ns object and remove finalizers block

```
# k get ns local-path-storage -o json > ns.json
```

1. Apply ns.json via API finalize endpoint

```
# kubectl proxy --port=8001 & curl -s -H "Content-Type: application/json" -X PUT --data-binary @ns.json \
> http://127.0.0.1:8001/api/v1/namespaces/local-path-storage/finalize
[2] 2697
error: listen tcp 127.0.0.1:8001: bind: Only one usage of each socket address (protocol/network
```

address/port) is normally permitted.


```
{
  "kind": "Namespace",
  "apiVersion": "v1",
  "metadata": {
    "name": "local-path-storage",
    "uid": "4bd4cf19-8b37-49f5-a78c-9692a1ff6062",
    "resourceVersion": "20211",
    "creationTimestamp": "2025-09-14T17:36:15Z",
    "deletionTimestamp": "2025-09-15T15:10:14Z",
    "labels": {
      "kubernetes.io/metadata.name": "local-path-storage"
    },
    "annotations": {
      "kubectl.kubernetes.io/last-applied-configuration": "{\"apiVersion\":\"v1\",\"kind\":\"Namespace
\",\"metadata\":{\"annotations\":{},\"name\":\"local-path-storage\"}}\n"
    },
    "managedFields": [
      {
        "manager": "kubectl-client-side-apply",
        "operation": "Update",
        "apiVersion": "v1",
        "time": "2025-09-14T17:36:15Z",
        "fieldsType": "FieldsV1",
        "fieldsV1": {
          "f:metadata": {
            "f:annotations": {
              ".": {},
              "f:kubectl.kubernetes.io/last-applied-configuration": {}
            },
            "f:labels": {
              ".": {},
              "f:kubernetes.io/metadata.name": {}
            }
          }
        }
      },
      {
        "manager": "k3s",
        "operation": "Update",
        "apiVersion": "v1",
        "time": "2025-09-15T15:10:21Z",
        "fieldsType": "FieldsV1",
        "fieldsV1": {
          "f:status": {
            "f:conditions": {
              ".": {},
              "k:{\"type\":\"NamespaceContentRemaining\"}": {
                ".": {},
                "f:lastTransitionTime": {},
                "f:message": {},
                "f:reason": {},
                "f:status": {},
```

```
            "f:type": {}
           },
           "k:{\"type\":\"NamespaceDeletionContentFailure\"}": {
            ".": {},
            "f:lastTransitionTime": {},
            "f:message": {},
            "f:reason": {},
            "f:status": {},
            "f:type": {}
           },
           "k:{\"type\":\"NamespaceDeletionDiscoveryFailure\"}": {
            ".": {},
            "f:lastTransitionTime": {},
            "f:message": {},
            "f:reason": {},
            "f:status": {},
            "f:type": {}
           },
           "k:{\"type\":\"NamespaceDeletionGroupVersionParsingFailure\"}": {
            ".": {},
            "f:lastTransitionTime": {},
            "f:message": {},
            "f:reason": {},
            "f:status": {},
            "f:type": {}
           },
           "k:{\"type\":\"NamespaceFinalizersRemaining\"}": {
            ".": {},
            "f:lastTransitionTime": {},
            "f:message": {},
            "f:reason": {},
            "f:status": {},
            "f:type": {}
           }
          }
         }
        },
        "subresource": "status"
       }
      ]
     },
     "spec": {},
     "status": {
      "phase": "Terminating",
      "conditions": [
       {
        "type": "NamespaceDeletionDiscoveryFailure",
        "status": "True",
        "lastTransitionTime": "2025-09-15T15:10:19Z",
        "reason": "DiscoveryFailed",
        "message": "Discovery failed for some groups, 1 failing: unable to retrieve the complete list of
server APIs: metrics.k8s.io/v1beta1: stale GroupVersion discovery: metrics.k8s.io/v1beta1"
       },
```

```
    {
      "type": "NamespaceDeletionGroupVersionParsingFailure",
      "status": "False",
      "lastTransitionTime": "2025-09-15T15:10:19Z",
      "reason": "ParsedGroupVersions",
      "message": "All legacy kube types successfully parsed"
    },
    {
      "type": "NamespaceDeletionContentFailure",
      "status": "False",
      "lastTransitionTime": "2025-09-15T15:10:21Z",
      "reason": "ContentDeleted",
      "message": "All content successfully deleted, may be waiting on finalization"
    },
    {
      "type": "NamespaceContentRemaining",
      "status": "False",
      "lastTransitionTime": "2025-09-15T15:10:19Z",
      "reason": "ContentRemoved",
      "message": "All content successfully removed"
    },
    {
      "type": "NamespaceFinalizersRemaining",
      "status": "False",
      "lastTransitionTime": "2025-09-15T15:10:19Z",
      "reason": "ContentHasNoFinalizers",
      "message": "All content-preserving finalizers finished"
    }
  ]
 }
}[2]+  Exit 1                  kubectl proxy --port=8001
```

NS was gone after.

1. Reapply manifest

```
# kubectl apply -f https://raw.githubusercontent.com/rancher/local-path-provisioner/master/deploy/local-path-storage.yaml

namespace/local-path-storage created
serviceaccount/local-path-provisioner-service-account created
role.rbac.authorization.k8s.io/local-path-provisioner-role created
clusterrole.rbac.authorization.k8s.io/local-path-provisioner-role unchanged
rolebinding.rbac.authorization.k8s.io/local-path-provisioner-bind created
clusterrolebinding.rbac.authorization.k8s.io/local-path-provisioner-bind unchanged
deployment.apps/local-path-provisioner created
storageclass.storage.k8s.io/local-path unchanged
configmap/local-path-config created
```

1. Check local-path-storage pods are up (previously they were not coming up)

```
# k get all -n local-path-storage
NAME                                     READY   STATUS    RESTARTS   AGE
pod/local-path-provisioner-7d6dddf9dd-8ngmq   1/1     Running   0          47s
```

```
NAME                          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/local-path-provisioner   1/1     1            1           47s

NAME                                    DESIRED   CURRENT   READY   AGE
replicaset.apps/local-path-provisioner-7d6dddf9dd   1         1         1       47s
```

1. Check sc, pvc, pv all are up (now they are all there! Previously they weren't)

```
#  k get sc
NAME                   PROVISIONER             RECLAIMPOLICY   VOLUMEBINDINGMODE
ALLOWVOLUMEEXPANSION   AGE
local-path (default)   rancher.io/local-path   Delete          WaitForFirstConsumer   false                  7d5h

# k get pvc
NAME                        STATUS   VOLUME                                     CAPACITY   ACCESS MODES
STORAGECLASS   VOLUMEATTRIBUTESCLASS   AGE
postgres-storage-postgres-0   Bound    pvc-bab4c32a-5989-4662-ac1a-5f32c2f6495d   1Gi        RWO
local-path     <unset>                 22h

# k get pv
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM
STORAGECLASS   VOLUMEATTRIBUTESCLASS   REASON   AGE
pvc-bab4c32a-5989-4662-ac1a-5f32c2f6495d   1Gi        RWO            Delete           Bound
default/postgres-storage-postgres-0   local-path     <unset>                          14h
```

1. Check postgres Pod

It has ContainerConfigError

```
# k get po | grep postgres -i
postgres-0                  0/1     CreateContainerConfigError   0          30s
postgres-76d88db5cd-wxm6h   1/1     Running                      2 (139m ago)   7d2h


Events:
  Type     Reason     Age              From               Message
  ----     ------     ----             ----               -------
  Normal   Scheduled  42s              default-scheduler  Successfully assigned default/postgres-0 to
worker1
  Normal   Pulled     13s (x5 over 42s)  kubelet          Container image "postgres:15" already present on
machine
  Warning  Failed     13s (x5 over 42s)  kubelet          Error: secret "postgres-secret" not found
```

1. Create secret and verify

```
# kubectl create secret generic postgres-secret \
> --from-literal=password=admin123
secret/postgres-secret created

# k get secret postgres-secret -oyaml
apiVersion: v1
data:
  password: YWRtaW4xMjM=
kind: Secret
```

```
metadata:
  creationTimestamp: "2025-09-15T15:57:28Z"
  name: postgres-secret
  namespace: default
  resourceVersion: "21865"
  uid: 0372ff80-5862-444d-837a-b5718e7a7a9d
type: Opaque
```

*The Postgres STS container gets its initial credentials from postgres-secret for Postgres itself (at container startup)
When Postgres first runs, it bootstraps the db with these values (creates the user and db)
This is not about apps connecting through a service - it's literally how Postgres knows its admin username, pw and which db to initialize.

K8s injects the secret into the container env, so postgres pod takes the value from the Secret's POSTGRES_PASSWORD.

- For POSTGRES_USER and POSTGRES_DB, you can:
  - Define them explicitly in the STS and point them to keys in the Secret OR
  - Let Postgres default them (*postgres* user, db *postgres* if nothing is set)

1. Restart pod and now pod is running FINALLY!

```
# k get po | grep -i postgres
postgres-0                  1/1     Running     0              18s
postgres-76d88db5cd-wxm6h   1/1     Running     2 (144m ago)   7d2h
```

Troubleshooting Log 3

Sunday, 28 September 2025 11:15 pm

*****ALL BELOW ISSUES ARE FROM INGRESS SECTION.**

6 ISSUES

ISSUE 1 : Command hung and already made resources in first try prevented second try from continuing, with error shown:
Error: INSTALLATION FAILED: Unable to continue with install: ClusterRole "ingress-nginx" in namespace "" exists and cannot be imported into the current release: invalid ownership metadata; annotation validation error: key "meta.helm.sh/release-namespace" must equal "ingress-nginx": current value is "default"

FIX: Delete already made resources

```
# kubectl delete
clusterrole,clusterrolebinding,serviceaccount,configmap,service,deployment,daemonset --
selector=app.kubernetes.io/name=ingress-nginx -A

# kubectl get all -A | grep ingress-nginx    (there was 1 pod stuck in terminating state)

# k delete po ingress-nginx-controller-6b96cf684f-n2sd7  -n default --force
# k get ingressclass
# k delete ingressclass nginx

# k get ingressclass
# k get all | grep -i ingress-nginx-admission
# kubectl get validatingwebhookconfigurations.admissionregistration.k8s.io | grep ingress
# kubectl delete validatingwebhookconfiguration ingress-nginx-admission
```

*Run install command again

1. Verify

```
# helm list -n ingress-nginx
NAME          NAMESPACE      REVISION       UPDATED                          STATUS         CHART
APP VERSION
ingress-nginx   ingress-nginx   1           2025-09-18 20:00:45.3931897 +0800 +08   deployed
ingress-nginx-4.13.2    1.13.2
```

*ingress-nginx Helm release shld be **deployed**

```
# k get po -n ingress-nginx -owide
NAME                                READY   STATUS    RESTARTS   AGE    IP             NODE     NOMINATED
NODE   READINESS GATES
ingress-nginx-controller-6b96cf684f-d69f8   1/1    Running   0       3h17m   172.18.137.126   master1
<none>         <none>

# k get svc -n ingress-nginx
NAME                       TYPE         CLUSTER-IP     EXTERNAL-IP   PORT(S)                AGE
ingress-nginx-controller          LoadBalancer   10.43.230.85    <pending>
```

```
80:30644/TCP,443:30589/TCP   3h27m
ingress-nginx-controller-admission   ClusterIP     10.43.251.154   <none>       443/TCP
3h27m

# k get ingressclass
NAME     CONTROLLER                  PARAMETERS   AGE
nginx    k8s.io/ingress-nginx        <none>       3h28m
traefik  traefik.io/ingress-controller   <none>      10d

# k get validatingwebhookconfigurations | grep ingress
ingress-nginx-admission   1         3h28m
```

2. Test with app service
   Create sampapp-ingress.yaml

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: sampapp-ingress
  namespace: default
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  ingressClassName: nginx
  rules:
  - host: sampapp.local
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: sampapp-service
            port:
              number: 80
```

   kubectl apply -f sampapp-ingress.yaml

3. Map host locally since external IP is pending
   Edit /etc/hosts (I added the bold line)

```
# Added by Docker Desktop
192.168.1.9 host.docker.internal
192.168.1.9 gateway.docker.internal
# To allow the same kube context to work on the host and the container:
127.0.0.1 kubernetes.docker.internal
127.0.0.1 sampapp.local
# End of section
```

4. Test connectivity

```
# curl -H "Host: sampapp.local" http://192.168.81.128:30644
<html>
<head><title>503 Service Temporarily Unavailable</title></head>
```

```
<body>
<center><h1>503 Service Temporarily Unavailable</h1></center>
<hr><center>nginx</center>
</body>
</html>
```

*This 503 is a good sign - means request is reaching the ingress-nginx controller but just can't find a backend svc to route to.
So ingress-controller is working fine. The problem is the Ingress or Service wiring.

ISSUE 2 : Error 503 (Either ingress or svc issue)

```
# kubectl get ingress -n default
NAME CLASS HOSTS ADDRESS PORTS AGE sampapp-ingress nginx sampapp.local 80 6m24s

# kubectl describe ingress sampapp-ingress -n default
Name:           sampapp-ingress
Labels:         <none>
Namespace:      default
Address:
Ingress Class:  nginx
Default backend:  <default>
Rules:
  Host        Path  Backends
  ----        ----  --------
  sampapp.local
              /   sampapp-service:80 (172.18.137.65:5000,172.18.137.70:5000)
Annotations:    nginx.ingress.kubernetes.io/rewrite-target: /
Events:
  Type    Reason  Age    From                      Message
  ----    ------  ----   ----                      -------
  Normal  Sync    6m42s  nginx-ingress-controller  Scheduled for sync
```

**Check that under rules, it shows sampapp.local, and that it says which service and port it's routing to

```
# kubectl get svc -n default
NAME            TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
kubernetes      ClusterIP   10.43.0.1       <none>        443/TCP    10d
nginx-service   ClusterIP   10.43.104.188   <none>        80/TCP     10d
postgres-service  ClusterIP  10.43.178.155  <none>        5432/TCP   10d
redis-service   ClusterIP   10.43.146.71    <none>        6379/TCP   10d
sampapp-service  ClusterIP  10.43.133.187   <none>        5000/TCP   10d
```

*From here we can see that there is a port mismatch for sampapp-service (service exposing 5000 vs ingress routing to 80)

*Edit sampapp-ingress.yaml so that the port is 5000 - matching svc

Test again

```
# kubectl get ingress -n default
NAME            CLASS   HOSTS          ADDRESS   PORTS   AGE
sampapp-ingress   nginx   sampapp.local            80      12m
```

```
# kubectl describe ingress sampapp-ingress -n default
Name:             sampapp-ingress
Labels:           <none>
Namespace:        default
Address:
Ingress Class:    nginx
Default backend:  <default>
Rules:
  Host         Path  Backends
  ----         ----  --------
  sampapp.local
               /   sampapp-service:5000 (172.18.137.65:5000,172.18.137.70:5000)
Annotations:     nginx.ingress.kubernetes.io/rewrite-target: /
Events:
  Type    Reason  Age              From                    Message
  ----    ------  ----             ----                    -------
  Normal  Sync    54s (x2 over 12m)  nginx-ingress-controller  Scheduled for sync

# kubectl get svc -n default
NAME              TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
kubernetes        ClusterIP   10.43.0.1       <none>        443/TCP    10d
nginx-service     ClusterIP   10.43.104.188   <none>        80/TCP     10d
postgres-service  ClusterIP   10.43.178.155   <none>        5432/TCP   10d
redis-service     ClusterIP   10.43.146.71    <none>        6379/TCP   10d
sampapp-service   ClusterIP   10.43.133.187   <none>        5000/TCP   10d

# curl -H "Host: sampapp.local" http://192.168.81.128:30644
<html>
<head><title>504 Gateway Time-out</title></head>
<body>
<center><h1>504 Gateway Time-out</h1></center>
<hr><center>nginx</center>
</body>
</html>
```

503 error solved.

ISSUE 3 : 504 error gateway timeout

```
# k get ep sampapp-service -n default -owide
Warning: v1 Endpoints is deprecated in v1.33+; use discovery.k8s.io/v1 EndpointSlice
NAME              ENDPOINTS                           AGE
sampapp-service   172.18.137.65:5000,172.18.137.70:5000   10d
*It shld list pod Ips and ports (5000). If it is <none> then svc selector doesn't match pod labels
```

```
# kubectl describe svc sampapp-service -n default
Name:                sampapp-service
Namespace:           default
Labels:              <none>
Annotations:         <none>
Selector:            app=sampapp
Type:                ClusterIP
IP Family Policy:    SingleStack
```

```
IP Families:            IPv4
IP:                     10.43.133.187
IPs:                    10.43.133.187
Port:                   <unset>  5000/TCP
TargetPort:             5000/TCP
Endpoints:              172.18.137.65:5000,172.18.137.70:5000
Session Affinity:       None
Internal Traffic Policy:  Cluster
Events:                 <none>
```

```
# kubectl get pods -n default --show-labels
NAME                    READY   STATUS              RESTARTS       AGE     LABELS
busybox                 1/1     Running             16 (30m ago)   10d     app=busybox
netshoot                1/1     Running             11 (29m ago)   3d21h   app=netshoot
nginx-96b9d695-4rwwl    0/1     Completed           0              3h53m   app=nginx,pod-template-
hash=96b9d695
nginx-96b9d695-64pkw    0/1     Evicted             0              3h53m   app=nginx,pod-template-
hash=96b9d695
nginx-96b9d695-bmgqq    0/1     Evicted             0              3h53m   app=nginx,pod-template-
hash=96b9d695
nginx-96b9d695-bwnqv    0/1     Evicted             0              3h53m   app=nginx,pod-template-
hash=96b9d695
nginx-96b9d695-c6r4z    0/1     Evicted             0              3h53m   app=nginx,pod-template-
hash=96b9d695
nginx-96b9d695-c7mmz    0/1     Evicted             0              3h53m   app=nginx,pod-template-
hash=96b9d695
nginx-96b9d695-d8gtc    0/1     Evicted             0              3h53m   app=nginx,pod-template-
hash=96b9d695
nginx-96b9d695-f4n9z    0/1     Evicted             0              3h53m   app=nginx,pod-template-
hash=96b9d695
nginx-96b9d695-f5sjg    0/1     Evicted             0              3h53m   app=nginx,pod-template-
hash=96b9d695
nginx-96b9d695-ggf96    0/1     Evicted             0              3h53m   app=nginx,pod-template-
hash=96b9d695
nginx-96b9d695-hdch7    0/1     Evicted             0              3h53m   app=nginx,pod-template-
hash=96b9d695
nginx-96b9d695-jkfrf    0/1     Evicted             0              3h53m   app=nginx,pod-template-
hash=96b9d695
nginx-96b9d695-jt7v7    1/1     Running             0              3h41m   app=nginx,pod-template-
hash=96b9d695
nginx-96b9d695-klhfp    0/1     Evicted             0              3h53m   app=nginx,pod-template-
hash=96b9d695
nginx-96b9d695-n2jp6    0/1     Evicted             0              3h53m   app=nginx,pod-template-
hash=96b9d695
nginx-96b9d695-njl7q    0/1     Evicted             0              3h53m   app=nginx,pod-template-
hash=96b9d695
nginx-96b9d695-nxxxg    0/1     Evicted             0              3h53m   app=nginx,pod-template-
hash=96b9d695
nginx-96b9d695-q6zz6    0/1     Evicted             0              3h53m   app=nginx,pod-template-
hash=96b9d695
nginx-96b9d695-s7p2m    0/1     Evicted             0              3h53m   app=nginx,pod-template-
hash=96b9d695
nginx-96b9d695-s9925    0/1     Evicted             0              3h53m   app=nginx,pod-template-
```

```
hash=96b9d695
nginx-96b9d695-srrdm       0/1     Evicted                 0              3h53m   app=nginx,pod-template-
hash=96b9d695
nginx-96b9d695-th6f5       0/1     Evicted                 0              3h53m   app=nginx,pod-template-
hash=96b9d695
nginx-96b9d695-whw7d       0/1     ContainerStatusUnknown   3 (4h48m ago)   3d14h
app=nginx,pod-template-hash=96b9d695
nginx-96b9d695-zfcwx       0/1     Evicted                 0              3h53m   app=nginx,pod-template-
hash=96b9d695
pg-backup-29301240-wc98k   0/1     Completed               0              17h
batch.kubernetes.io/controller-uid=56d446d1-5e88-4700-94e1-
d82f26ef288e,batch.kubernetes.io/job-name=pg-backup-29301240,controller-uid=
56d446d1-5e88-4700-94e1-d82f26ef288e,job-name=pg-backup-29301240
pg-backup-29302680-wzzjp   0/1     Completed               0              4h48m
batch.kubernetes.io/controller-uid=908fb3b7-85ca-41f9-b500-
d9d320816056,batch.kubernetes.io/job-name=pg-backup-29302680,controller-uid=
908fb3b7-85ca-41f9-b500-d9d320816056,job-name=pg-backup-29302680
pg-backup-test-74hj4       0/1     Completed               0              2d23h
batch.kubernetes.io/controller-uid=cfc0aeba-691a-4dad-a51a-
a99c5c521895,batch.kubernetes.io/job-name=pg-backup-test,controller-
uid=cfc0aeba-691a-4dad-a51a-a99c5c521895,job-name=pg-backup-test
postgres-0                 1/1     Running                 0              3h41m
app=postgres,apps.kubernetes.io/pod-index=0,controller-revision-
hash=postgres-5748f9c956,statefulset.kubernetes.io/pod-name=postgres-0
postgres-76d88db5cd-v9qdj  1/1     Running                 0              3h41m   app=postgres,pod-
template-hash=76d88db5cd
postgres-76d88db5cd-vwt68  0/1     Completed               0              3h53m   app=postgres,pod-
template-hash=76d88db5cd
postgres-76d88db5cd-wxm6h  0/1     Completed               4 (4h48m ago)   10d
app=postgres,pod-template-hash=76d88db5cd
redis-7b986b9f57-b4jtm     1/1     Running                 0              3h41m   app=redis,pod-template-
hash=7b986b9f57
redis-7b986b9f57-jsh2q     0/1     Completed               4 (4h48m ago)   10d     app=redis,pod-
template-hash=7b986b9f57
sampapp-6788976bbf-26f8x   0/2     Evicted                 0              3h41m   app=sampapp,pod-
template-hash=6788976bbf
sampapp-6788976bbf-6l2pl   0/2     Error                   8              10d     app=sampapp,pod-
template-hash=6788976bbf
sampapp-6788976bbf-ftzc8   0/2     Evicted                 0              3h41m   app=sampapp,pod-
template-hash=6788976bbf
sampapp-6788976bbf-g6797   0/2     Error                   8              10d     app=sampapp,pod-
template-hash=6788976bbf
sampapp-6788976bbf-hvhvq   0/2     Evicted                 0              3h53m   app=sampapp,pod-
template-hash=6788976bbf
sampapp-6788976bbf-jpj4w   0/2     Evicted                 0              3h41m   app=sampapp,pod-
template-hash=6788976bbf
sampapp-6788976bbf-lrfjs   0/2     Evicted                 0              3h53m   app=sampapp,pod-
template-hash=6788976bbf
sampapp-6788976bbf-n9tjk   2/2     Running                 0              3h41m   app=sampapp,pod-
template-hash=6788976bbf
sampapp-6788976bbf-njdgl   0/2     ContainerStatusUnknown   2              3h53m
app=sampapp,pod-template-hash=6788976bbf
sampapp-6788976bbf-pcpxr   0/2     Evicted                 0              3h41m   app=sampapp,pod-
```

```
template-hash=6788976bbf
sampapp-6788976bbf-qpbnh   0/2   Evicted            0          3h41m   app=sampapp,pod-
template-hash=6788976bbf
sampapp-6788976bbf-r22lc   2/2   Running            0          3h58m   app=sampapp,pod-
template-hash=6788976bbf
sampapp-6788976bbf-swbv7   0/2   Evicted            0          3h53m   app=sampapp,pod-
template-hash=6788976bbf
sampapp-6788976bbf-tmpm7   0/2   Evicted            0          3h41m   app=sampapp,pod-
template-hash=6788976bbf
sampapp-6788976bbf-v5pf6   0/2   Evicted            0          3h53m   app=sampapp,pod-
template-hash=6788976bbf
sampapp-6788976bbf-vfwx9   0/2   Evicted            0          3h53m   app=sampapp,pod-
template-hash=6788976bbf
sampapp-6788976bbf-vmrmf   0/2   Evicted            0          3h41m   app=sampapp,pod-
template-hash=6788976bbf
sampapp-6788976bbf-zzfqq   0/2   Evicted            0          3h41m   app=sampapp,pod-
template-hash=6788976bbf
test1                      0/1   Completed          0          3d1h    run=test1
test2                      0/1   Completed          0          3d1h    run=test2
```

*EP and svc are OK, but there're so many failed pods present (sampapp) so Ingress mightve briefly routed traffic to the evicted pods causing 504s.

Delete all evicted pods

```
# kubectl get pods --all-namespaces | grep Evicted | awk '{print $2 " -n " $1}' | xargs -r kubectl delete pod    (somehow didn’t delete default ns pods)
# kubectl get pods -n default | grep Evicted | awk '{print $1}' | xargs -r kubectl delete pod -n default
```

Check that only left with the 2 running pods

```
# kubectl get pods -n default -o wide | grep sampapp
sampapp-6788976bbf-n9tjk   2/2   Running            0          3h53m   172.18.137.70   master1
<none>          <none>
sampapp-6788976bbf-r22lc   2/2   Running            0          4h10m   172.18.137.65   master1
<none>          <none>
```

Check that ep point only to those 2 running pods

```
# kubectl get endpoints sampapp-service -n default -o wide
Warning: v1 Endpoints is deprecated in v1.33+; use discovery.k8s.io/v1 EndpointSlice
NAME              ENDPOINTS                                 AGE
sampapp-service   172.18.137.65:5000,172.18.137.70:5000   10d
```

Check ingress wiring

```
# kubectl describe ingress sampapp-ingress -n default
Name:             sampapp-ingress
Labels:           <none>
Namespace:        default
Address:
Ingress Class:    nginx
Default backend:  <default>
Rules:
```

```
 Host        Path  Backends
 ----        ----  --------
 sampapp.local
             /   sampapp-service:5000 (172.18.137.65:5000,172.18.137.70:5000)
Annotations:    nginx.ingress.kubernetes.io/rewrite-target: /
Events:
 Type    Reason  Age            From                  Message
 ----    ------  ----           ----                  -------
 Normal  Sync    21m (x2 over 32m)  nginx-ingress-controller  Scheduled for sync
```

*Shld point to sampapp-service:5000 and events shld be "Scheduled for sync" without errors

Test curl again

```
# curl -H "Host: sampapp.local" http://192.168.81.128:30644
```

Still 504 error

ISSUE 4 Ingress svc external IP pending, svc loadbalancer helpers failing, local-path-provisioner is broken

2 options to fix :

- Quick workaround (use NodePort directly)
- Proper fix (install MetalLB for bare metal)

1. For the purpose of this lab, I chose option 1

```
# k edit svc ingress-nginx-controller -n ingress-nginx
Changed:
        type: LoadBalancer -> type: NodePort
```

2. Saved and checked that below has changed to NodePort:

```
# kubectl get svc -n ingress-nginx
NAME                       TYPE      CLUSTER-IP     EXTERNAL-IP   PORT(S)                      AGE
ingress-nginx-controller   NodePort  10.43.230.85   <none>
80:30644/TCP,443:30589/TCP   9h
ingress-nginx-controller-admission   ClusterIP   10.43.251.154   <none>      443/TCP
9h
```

Still got 504 Gateway Timeout error.

3. Checked connectivity from inside ingress-controller pod

```
# kubectl -n ingress-nginx exec -it deploy/ingress-nginx-controller -- curl -s http://sampapp-
service.default.svc.cluster.local:5000
```

It hung

ISSUE 5 Nodes unhealthy (CPU and disk)

1. Checked nodes health

```
# df -h              -- was 95% for / dir
# top                -- CPU was 3++ (nproc is only 2)
```

2. Cleared with:

```
# docker system prune -af --volumes    (clear old images)
```

```
# crictl rmi --prune
# journalctl --vacuum-time=2d
```

3. After clearing, both cpu and fs usage went down to healthy levels on node (via df -h and cpu), but checking from cluster, worker1 still has DiskPressure:

```
# kubectl describe node master1 | grep -A5 Conditions
Conditions:
  Type             Status  LastHeartbeatTime             LastTransitionTime            Reason
Message
  ----             ------  -----------------             ------------------            ------                 -------
  NetworkUnavailable   False   Fri, 19 Sep 2025 05:52:39 +0800   Fri, 19 Sep 2025 05:52:39 +0800
CalicoIsUp               Calico is running on this node
  MemoryPressure       False   Fri, 19 Sep 2025 06:09:30 +0800   Mon, 08 Sep 2025 18:09:36 +0800
KubeletHasSufficientMemory   kubelet has sufficient memory available
  DiskPressure         False   Fri, 19 Sep 2025 06:09:30 +0800   Fri, 19 Sep 2025 06:08:49 +0800
KubeletHasNoDiskPressure     kubelet has no disk pressure
```

```
# kubectl describe node worker1 | grep -A5 Conditions
Conditions:
  Type             Status  LastHeartbeatTime             LastTransitionTime            Reason
Message
  ----             ------  -----------------             ------------------            ------                 -------
  NetworkUnavailable   False   Fri, 19 Sep 2025 05:49:42 +0800   Fri, 19 Sep 2025 05:49:42 +0800
CalicoIsUp               Calico is running on this node
  MemoryPressure       False   Fri, 19 Sep 2025 06:09:08 +0800   Mon, 08 Sep 2025 18:14:07 +0800
KubeletHasSufficientMemory   kubelet has sufficient memory available
  DiskPressure         True    Fri, 19 Sep 2025 06:09:08 +0800   Fri, 19 Sep 2025 06:07:16 +0800
KubeletHasDiskPressure       kubelet has disk pressure
```

^^This is because Kubernetes doesn't only check root usage. It tracks ephemeral storage (container writable layers, logs, emptyDirs, etc) under /var/lib/kubelet, /var/lib/containerd, or /var/lib/docker

Deleted all Failed pods

```
# kubectl get pods -A --field-selector=status.phase=Failed -o jsonpath='{range .items[*]}
{.metadata.namespace}{" "}{.metadata.name}{"\n"}{end}' \
| xargs -r -n2 sh -c 'kubectl delete pod -n "$0" "$1"'
```

Checked worker1 again and it had no disk pressure

```
# kubectl describe no worker1 | grep -A5 Conditions
Conditions:
  Type             Status  LastHeartbeatTime             LastTransitionTime            Reason
Message
  ----             ------  -----------------             ------------------            ------                 -------
  NetworkUnavailable   False   Fri, 19 Sep 2025 05:49:42 +0800   Fri, 19 Sep 2025 05:49:42 +0800
CalicoIsUp               Calico is running on this node
  MemoryPressure       False   Fri, 19 Sep 2025 06:19:08 +0800   Mon, 08 Sep 2025 18:14:07 +0800
KubeletHasSufficientMemory   kubelet has sufficient memory available
  DiskPressure         False   Fri, 19 Sep 2025 06:19:08 +0800   Fri, 19 Sep 2025 06:14:03 +0800
KubeletHasNoDiskPressure     kubelet has no disk pressure
```

ISSUE 5 FIXED

ISSUE 6 NETPOL

1. Test Ingress from inside cluster

```
# kubectl run curltest --rm -it --image=curlimages/curl --restart=Never -- \
  curl http://sampapp-service.default.svc.cluster.local:5000
```

And from outside

```
# curl -H "Host: sampapp.local" http://<node-ip>:<nodeport>
```

Curl was still not working!

The **504 Gateway Timeout** means:
Ingress controller received your request → tried to connect to sampapp-service:5000 → could not get a response in time.
So the real suspects are still:

1. **App pod binding issue** → app is listening on 127.0.0.1:5000 instead of 0.0.0.0:5000.
    - Run:

      kubectl logs -n default <sampapp-pod> -c sampapp

      Look for the bind address.
2. **Service selector mismatch** → but you already confirmed endpoints exist, so selector is fine.
3. **App crash loop or stuck** → check if the logs show errors when requests arrive.
4. **Ingress timeout settings** → default is 60s, but your curl fails fast, which means the backend isn't reachable at all.

Check for bind address - first address line 0.0.0.0 is correct.

```
# kubectl logs -n default sampapp-6788976bbf-l98p6 -c sampapp
WARNING: This is a development server. Do not use it in a production deployment. Use a production
WSGI server instead.
 * Running on all addresses (0.0.0.0)     (bind add)
 * Running on http://127.0.0.1:5000      (loopback inside container)
 * Running on http://172.18.137.94:5000    (pod's actual IP)
Press CTRL+C to quit
 * Serving Flask app 'app'
 * Debug mode: off
```

Tried connectivity from ingress-nginx to pod again

```
# kubectl exec -it -n ingress-nginx <ingress-nginx-pod-name> -- curl -s -H "Host: sampapp.local"
http://sampapp-service.default.svc.cluster.local:5000
```

It hung. And suddenly it hit me!!! I rmb I set some test netpols for calico earlier… and these were blocking my traffic omg
Allow netshoot > sampapp

I deleted all netpols (will look at proper implementation later on)

And now connectivity is working…

```
# curl -H "Host: sampapp.local" http://192.168.81.128:30644
<html>
```

```
<head>
  <title>Feedback Form</title>
  <link rel="stylesheet" href="/static/style.css">
</head>
<body>
  <div class="container">
    <h2>Feedback Form</h2>
    <form method="POST">
      <input type="text" name="name" placeholder="Your Name" required>
      <input type="email" name="email" placeholder="Your Email" required>
      <textarea name="message" placeholder="Your Feedback" required></textarea>
      <button type="submit">Submit</button>
    </form>
  </div>
</body>
</html>
```

Those netpols messed with me twice now… need to be more mindful

Troubleshooting Log 4

Sunday, 28 September 2025 11:24 pm

*****ALL BELOW ISSUES ARE FROM CI/CD w JENKINS & ARGOCD SECTION.**

12 ISSUES

ISSUE 1 EPHEMERAL STORAGE OF NODES CAUSING ALL PODS TO FAIL

After ArgoCD, all pods in cluster started to fail. With errors hinting at ephemeral storage issues. Looking at each node, the disk space on the node itself was high and CPU was going haywire (above cpu cores)

```
root@master1:~# du -sh /var/lib/rancher/k3s/agent/containerd/
3.5G /var/lib/rancher/k3s/agent/containerd/

root@master1:~# df -h /
Filesystem Size Used Avail Use% Mounted on
/dev/mapper/ubuntu--vg-ubuntu--lv 9.8G 8.3G 970M 90% /
```

90% usage in / and /var/lib/rancher/k3s/agent/containerd alone is 3.5G - kubelet sees less than 1G free and immediately applies pressure

1. Expand virtual disk in Vmware Workstation
   - Power off both VMs
   - In Vmware Workstation, Click on Hard Disk > Expand
   - Current: 20GB > 40GB
2. Boot the VM and check new disk size
   - Check raw disk size

```
miki@master1:~$ lsblk
NAME                      MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
loop0                       7:0    0 63.8M  1 loop /snap/core20/2599
loop1                       7:1    0 63.9M  1 loop /snap/core20/2318
loop2                       7:2    0   87M  1 loop /snap/lxd/29351
loop3                       7:3    0 89.4M  1 loop /snap/lxd/31333
loop4                       7:4    0 50.8M  1 loop /snap/snapd/25202
loop5                       7:5    0 49.3M  1 loop /snap/snapd/24792
sda                         8:0    0   40G  0 disk
 ├─sda1                     8:1    0    1M  0 part
 ├─sda2                     8:2    0  1.8G  0 part /boot
 └─sda3                     8:3    0 18.2G  0 part
   └─ubuntu--vg-ubuntu--lv 253:0    0   10G  0 lvm  /
sr0                        11:0    1 1024M  0 rom
```

   - Df -h will still show old root partition size since we haven't extended it

```
miki@master1:~$ df -h
Filesystem                         Size  Used Avail Use% Mounted on
tmpfs                              388M  3.0M  385M   1% /run
/dev/mapper/ubuntu--vg-ubuntu--lv  9.8G  7.7G  1.6G  84% /
tmpfs                              1.9G     0  1.9G   0% /dev/shm
tmpfs                              5.0M     0  5.0M   0% /run/lock
```

```
/dev/sda2                 1.8G  252M  1.4G  16% /boot
shm                       64M     0   64M   0%
/run/k3s/containerd/io.containerd.grpc.v1.cri/sandboxes/79e6e9de8c03dd811eaa879154b5954
ce7598d9cd313dc15949ba52d57292a70/shm
shm                       64M     0   64M   0%
/run/k3s/containerd/io.containerd.grpc.v1.cri/sandboxes/31b93d5f3931ef02b4cb39b298b0b874
c5fc9596281e8a6f88a764ce336759ef/shm
shm                       64M     0   64M   0%
/run/k3s/containerd/io.containerd.grpc.v1.cri/sandboxes/3f0577a71b8199ac0b3fbcad431ea4de
34cde190fc1593a7467be3e4521502e9/shm
shm                       64M     0   64M   0%
/run/k3s/containerd/io.containerd.grpc.v1.cri/sandboxes/45b627f1adf2372fe53249430094496e
244f3025a2a64de788d2c57704875457/shm
shm                       64M     0   64M   0%
/run/k3s/containerd/io.containerd.grpc.v1.cri/sandboxes/02d242e8cb36a8a4d53c6025c4afd7c1
c32858996f9ac1d7951e79085c4dd478/shm
shm                       64M   28K   64M   1%
/run/k3s/containerd/io.containerd.grpc.v1.cri/sandboxes/65b8efaae9924521ccf089034d3d8839
de0d7ad7a8710a3b575f5df6410a3e2f/shm
shm                       64M     0   64M   0%
/run/k3s/containerd/io.containerd.grpc.v1.cri/sandboxes/369918f2e218ff6ecb12fd6759d448582
a53d9c4ac6bad67b073f0d5912c4679/shm
shm                       64M     0   64M   0%
/run/k3s/containerd/io.containerd.grpc.v1.cri/sandboxes/a72764cb74b40cfc14f5823857e2722c2
ef415b025cd0f1a12e4520e11f7dd1b/shm
tmpfs                     388M  4.0K  388M   1% /run/user/1000
```

3. Extend the filesystem

   This is the pv, vg and lv before we do anything:

```
root@master1:~# pvs
  PV         VG        Fmt  Attr PSize  PFree
  /dev/sda3  ubuntu-vg lvm2 a--  18.22g 8.22g

root@master1:~# vgs
  VG        #PV #LV #SN Attr   VSize  VFree
  ubuntu-vg   1   1   0 wz--n- 18.22g 8.22g

root@master1:~# lvs
  LV        VG        Attr       LSize  Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  ubuntu-lv ubuntu-vg -wi-ao---- 10.00g
```

   We need to resize the partition first so that LVM can see the new space

```
# growpart /dev/sda 3
```

```
# pvresize /dev/sda3
```

   Now pvs shld show:

```
root@master1:~# pvs
  PV         VG        Fmt  Attr PSize  PFree
  /dev/sda3  ubuntu-vg lvm2 a--  38.22g 28.22g
```

   When I run vgs, it also shows up so no vgextend is required here.

```
root@master1:~# vgs
```

```
  VG       #PV #LV #SN Attr   VSize  VFree
  ubuntu-vg   1   1   0 wz--n- 38.22g 28.22g
```

Extend LV

```
(before) root@master1:~# lvs
  LV       VG        Attr       LSize  Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  ubuntu-lv ubuntu-vg -wi-ao---- 10.00g

(got the lvpath from df -h /)
root@master1:~# lvextend -l +100%FREE /dev/mapper/ubuntu--vg-ubuntu--lv
  Size of logical volume ubuntu-vg/ubuntu-lv changed from 10.00 GiB (2560 extents) to 38.22 GiB (9785 extents).
  Logical volume ubuntu-vg/ubuntu-lv successfully resized.

(after) root@master1:~# lvs
  LV       VG        Attr       LSize  Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  ubuntu-lv ubuntu-vg -wi-ao---- 38.22g
```

Resize fs

```
root@master1:~# resize2fs /dev/mapper/ubuntu--vg-ubuntu--lv
resize2fs 1.46.5 (30-Dec-2021)
Filesystem at /dev/mapper/ubuntu--vg-ubuntu--lv is mounted on /; on-line resizing required
old_desc_blocks = 2, new_desc_blocks = 5
The filesystem on /dev/mapper/ubuntu--vg-ubuntu--lv is now 10019840 (4k) blocks long.
```

Check new fs size

```
root@master1:~# df -h /
Filesystem                         Size  Used Avail Use% Mounted on
/dev/mapper/ubuntu--vg-ubuntu--lv  38G  7.3G   29G  21% /
```

lovely.

Now we need to restore the cluster to its working state…

4. Check health of nodes

```
# kubectl describe no master1 | grep -A5 Conditions
Conditions:
  Type             Status  LastHeartbeatTime               LastTransitionTime              Reason
Message
  ----             ------  -----------------               ------------------              ------
  NetworkUnavailable   False   Sun, 21 Sep 2025 15:02:07 +0800   Sun, 21 Sep 2025 15:02:07 +0800
CalicoIsUp                  Calico is running on this node
  MemoryPressure       False   Sun, 21 Sep 2025 15:01:57 +0800   Mon, 08 Sep 2025 18:09:36 +0800
KubeletHasSufficientMemory   kubelet has sufficient memory available
  DiskPressure         False   Sun, 21 Sep 2025 15:01:57 +0800   Sat, 20 Sep 2025 07:57:24 +0800
KubeletHasNoDiskPressure     kubelet has no disk pressure

# kubectl describe no worker1 | grep -A5 Conditions
Conditions:
```

```
 Type             Status  LastHeartbeatTime             LastTransitionTime            Reason
Message
 ----             ------  -----------------             ------------------            ------                       -------
 NetworkUnavailable   False   Sun, 21 Sep 2025 15:02:13 +0800   Sun, 21 Sep 2025 15:02:13 +0800
CalicoIsUp                Calico is running on this node
 MemoryPressure       False   Sun, 21 Sep 2025 15:02:05 +0800   Mon, 08 Sep 2025 18:14:07 +0800
KubeletHasSufficientMemory   kubelet has sufficient memory available
 DiskPressure         False   Sun, 21 Sep 2025 15:02:05 +0800   Sat, 20 Sep 2025 08:03:39 +0800
KubeletHasNoDiskPressure     kubelet has no disk pressure
```

All are False… looking good

5. Check Calico Connectivity
   Had to recreate busybox and netshoot pods then test connectivity

```
# k exec -it netshoot -- sh

ping 8.8.8.8     (outbound internet check)
nslookup kubernetes.default   (DNS check)
ping <other-pod-IP      (pod to pod connectivity test)
curl -vk http://sampapp-service:5000      (service connectivity test)
```

Results:

```
# k exec -it netshoot -- sh
~ # ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=127 time=111 ms
nd64 bytes from 8.8.8.8: icmp_seq=2 ttl=127 time=135 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=127 time=53.2 ms
^C
--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 53.229/99.534/134.561/34.147 ms


# nslookup kubernetes.default
;; Got recursion not available from 10.43.0.10
;; Got recursion not available from 10.43.0.10
Server:         10.43.0.10
Address:        10.43.0.10#53

Name:   kubernetes.default.svc.cluster.local
Address: 10.43.0.1
;; Got recursion not available from 10.43.0.10


#  k exec -it netshoot -- sh
~ # ping 172.18.235.187
PING 172.18.235.187 (172.18.235.187) 56(84) bytes of data.
64 bytes from 172.18.235.187: icmp_seq=1 ttl=63 time=0.185 ms
64 bytes from 172.18.235.187: icmp_seq=2 ttl=63 time=0.050 ms
64 bytes from 172.18.235.187: icmp_seq=3 ttl=63 time=0.063 ms
^C
```

```
--- 172.18.235.187 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2043ms
rtt min/avg/max/mdev = 0.050/0.099/0.185/0.060 ms



#  k exec -it netshoot -- curl -vk sampapp-service:5000
* Host sampapp-service:5000 was resolved.
* IPv6: (none)
* IPv4: 10.43.133.187
*   Trying 10.43.133.187:5000...
* Connected to sampapp-service (10.43.133.187) port 5000
* using HTTP/1.x
> GET / HTTP/1.1
> Host: sampapp-service:5000
> User-Agent: curl/8.14.1
> Accept: */*
>
* Request completely sent off
< HTTP/1.1 200 OK
< Server: Werkzeug/3.1.3 Python/3.10.18
< Date: Sun, 21 Sep 2025 07:20:53 GMT
< Content-Type: text/html; charset=utf-8
< Content-Length: 538
< Connection: close
<
<html>
<head>
   <title>Feedback Form</title>
   <link rel="stylesheet" href="/static/style.css">
</head>
<body>
   <div class="container">
      <h2>Feedback Form</h2>
      <form method="POST">
         <input type="text" name="name" placeholder="Your Name" required>
         <input type="email" name="email" placeholder="Your Email" required>
         <textarea name="message" placeholder="Your Feedback" required></textarea>
         <button type="submit">Submit</button>
      </form>
   </div>
</body>
* shutting down connection #0
</html>
```

Calico Connectivity is good.

6. Check Postgres Health
   Check PV

```
# k get pv
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM
STORAGECLASS   VOLUMEATTRIBUTESCLASS   REASON   AGE
pvc-87ab800a-aab9-48a8-a2ea-3d57ab5bd9cb   1Gi        RWO            Delete           Bound    default/pg-
```

```
backup-pvc              local-path     <unset>                  5d14h
pvc-a5cd1783-13d6-4928-94f9-5aa81e9f1aae   8Gi       RWO          Delete         Bound    jenkins/jenkins
local-path     <unset>                  2d7h
pvc-bab4c32a-5989-4662-ac1a-5f32c2f6495d   1Gi       RWO          Delete         Bound
default/postgres-storage-postgres-0   local-path     <unset>                  6d6h
```

Check PVC

```
$ k get pvc
NAME                    STATUS   VOLUME                                     CAPACITY   ACCESS MODES
STORAGECLASS   VOLUMEATTRIBUTESCLASS   AGE
pg-backup-pvc           Bound    pvc-87ab800a-aab9-48a8-a2ea-3d57ab5bd9cb   1Gi        RWO
local-path     <unset>                 5d15h
postgres-storage-postgres-0   Bound    pvc-bab4c32a-5989-4662-ac1a-5f32c2f6495d   1Gi        RWO
local-path     <unset>                 6d13h
```

Check Postgres pod

```
# k get po -n default | grep postgres
postgres-0              1/1     Running             1 (32m ago)   31h
postgres-76d88db5cd-6wpwj   1/1     Running             1 (32m ago)   31h
```

Exec into postgrespod to run simple query

```
# kubectl exec -it postgres-0 -- psql -U postgres -d postgres
psql (15.14 (Debian 15.14-1.pgdg13+1))
Type "help" for help.

postgres=#
postgres=# SELECT NOW();
              now
-------------------------------
 2025-09-21 07:32:23.203402+00
(1 row)

postgres=#
```

Created simple pg-client.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: pg-client
  namespace: default
spec:
  containers:
  - name: pg-client
    image: postgres:15
    command: ["sleep", "3600"]
  restartPolicy: Always
```

Test

```
# kubectl exec -it pg-client -- psql -h postgres-service -U postgres -d postgres -c "SELECT version();"
```

```
Password for user postgres:
                                                  version
-------------------------------------------------------------------------------------------------------------------------
 PostgreSQL 15.14 (Debian 15.14-1.pgdg13+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian
14.2.0-19) 14.2.0, 64-bit
(1 row)
```

ISSUE 2: LOCAL-PATH-PROVISIONER-PODS IN CRASHLOOPBACKOFF (RBAC)
Noticed the local-path-provisioner pods were in CrashLoopBackOff state.
I described the log and got this:

```
# kubectl -n local-path-storage logs local-path-provisioner-66776cdb94-scm46
time="2025-09-21T07:44:49Z" level=fatal msg="Error starting daemon: invalid empty flag helper-pod-
file and it also does not exist at ConfigMap local-path-storage/local-path-config with err: configmaps
\"local-path-config\" is forbidden: User \"system:serviceaccount:local-path-storage:local-path-
provisioner-service-account\" cannot get resource \"configmaps\" in API group \"\" in the namespace
\"local-path-storage\""
```

Above shows that the local-path-provisioner pod is failing because it doesn't have RBAC permission to read the local-path-config CM in its namespace.

But jn the postgres pods, PVCs, PVs all are working fine because the PVCs were already bound to the PVs before this local-path-provisioner pod failed. So all of those resources came up fine since theyre just using their existing PVs. The failure of this local-path-provisioner pod only affects if you are trying to create new PVCs.

Give service account the correct RBAC role:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: local-path-provisioner-role
rules:
  - apiGroups: [""]
    resources: ["nodes", "persistentvolumeclaims", "persistentvolumes", "configmaps"]
    verbs: ["get", "list", "watch", "create", "delete", "update", "patch"]

  - apiGroups: [""]
    resources: ["events"]
    verbs: ["create", "patch", "update"]

---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: local-path-provisioner-bind
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: local-path-provisioner-role
subjects:
  - kind: ServiceAccount
    name: local-path-provisioner-service-account
```

```
  namespace: local-path-storage
```

Apply manifest

```
# k apply -f local-path-provisioner-role.yaml
clusterrole.rbac.authorization.k8s.io/local-path-provisioner-role configured
clusterrolebinding.rbac.authorization.k8s.io/local-path-provisioner-bind configured
```

Delete the old pods and watch the new one come up

```
# k get po -n local-path-storage
NAME                                READY   STATUS    RESTARTS   AGE
local-path-provisioner-66776cdb94-c4nv4   1/1     Running   0          3s
```

7. Check cronjob

Check pvc pv are there

```
# k get pvc
NAME                  STATUS   VOLUME                                     CAPACITY   ACCESS MODES
STORAGECLASS   VOLUMEATTRIBUTESCLASS   AGE
pg-backup-pvc         Bound    pvc-87ab800a-aab9-48a8-a2ea-3d57ab5bd9cb   1Gi        RWO
local-path     <unset>                 5d15h
postgres-storage-postgres-0   Bound    pvc-bab4c32a-5989-4662-ac1a-5f32c2f6495d   1Gi        RWO
local-path     <unset>                 6d14h

# k get pv
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM
STORAGECLASS   VOLUMEATTRIBUTESCLASS   REASON   AGE
pvc-87ab800a-aab9-48a8-a2ea-3d57ab5bd9cb   1Gi        RWO            Delete           Bound    default/pg-
backup-pvc             local-path     <unset>                          5d15h
pvc-a5cd1783-13d6-4928-94f9-5aa81e9f1aae   8Gi        RWO            Delete           Bound    jenkins/jenkins
local-path     <unset>                          2d8h
pvc-bab4c32a-5989-4662-ac1a-5f32c2f6495d   1Gi        RWO            Delete           Bound
default/postgres-storage-postgres-0   local-path     <unset>                 6d6h
```

Create test job

```
# kubectl create job --from=cronjob/pg-backup pg-backup-test
```

Check that job was created and completed

```
# k get job
NAME             STATUS     COMPLETIONS   DURATION   AGE
pg-backup-29304120   Complete   1/1           2m34s      2d5h
pg-backup-29307000   Complete   1/1           14m        52m
pg-backup-test       Complete   1/1           3s         15s
```

8. Check Ingress

➢ Verify Ingress Controller

```
# kubectl get pods -n ingress-nginx -o wide
```

- Make sure ingress-nginx-controller is **Running** and Ready = 1/1.
- Check logs for errors:

```
# kubectl logs -n ingress-nginx deploy/ingress-nginx-controller
```

- Verify ingress resource

```
# kubectl get ingress -A
# kubectl describe ingress sampapp-ingress -n default
```

Confirm HOSTS = sampapp.local
Backend points to sampapp-service:5000

- Verify service + pods

I checked that sampapp-service endpoints matched sampapp pod Ips, Service has ClusterIP and port 5000, pods are running and container listening on port 5000

```
# kubectl get svc -n default sampapp-service

# k get endpoints sampapp-service
sampapp-service    172.18.137.71:5000,172.18.235.140:5000   12d

# k get po -owide | grep sampapp
sampapp-f6c65db94-l72nz     2/2     Running             0          53m     172.18.235.140   worker1
<none>          <none>
sampapp-f6c65db94-wwfg4     2/2     Running             0          45m     172.18.137.71    master1
<none>          <none>
```

- In-cluster curl

```
# kubectl exec -it -n ingress-nginx deploy/ingress-nginx-controller -- \
  curl -s -H "Host: sampapp.local" http://sampapp-service.default.svc.cluster.local:5000
<html>
<head>
   <title>Feedback Form</title>
   <link rel="stylesheet" href="/static/style.css">
</head>
<body>
   <div class="container">
      <h2>Feedback Form</h2>
      <form method="POST">
         <input type="text" name="name" placeholder="Your Name" required>
         <input type="email" name="email" placeholder="Your Email" required>
         <textarea name="message" placeholder="Your Feedback" required></textarea>
         <button type="submit">Submit</button>
      </form>
   </div>
</body>
</html>
```

- From workstation curl

```
curl -H "Host: sampapp.local" http://192.168.81.128:30644
<html>
<head>
   <title>Feedback Form</title>
   <link rel="stylesheet" href="/static/style.css">
</head>
<body>
```

```
  <div class="container">
    <h2>Feedback Form</h2>
    <form method="POST">
      <input type="text" name="name" placeholder="Your Name" required>
      <input type="email" name="email" placeholder="Your Email" required>
      <textarea name="message" placeholder="Your Feedback" required></textarea>
      <button type="submit">Submit</button>
    </form>
  </div>
</body>
</html>
```

ISSUE 3 : UNKNOWN SYNC STATUS

Due to netpol

Sync status kept showing Unknown:

```
# k get app -n argocd
NAME     SYNC STATUS   HEALTH STATUS
sampapp  Unknown       Healthy
```

Argocd repo server pod running

```
# k get po -n argocd | grep repo
argocd-repo-server-6fccc5759b-lz7hl             1/1     Running   1 (13m ago)   3d4h
```

DNS test to github

```
# k run -it --rm --image=busybox:1.28 dns-test --restart=Never -- nslookup github.com
Server:    10.43.0.10
Address 1: 10.43.0.10 kube-dns.kube-system.svc.cluster.local

Name:      github.com
Address 1: 20.205.243.166
pod "dns-test" deleted
```

Test dns from inside repo server pod

```
# k exec -n argocd -it argocd-repo-server-6fccc5759b-lz7hl -c argocd-repo-server -- sh
$ getent hosts github.com
20.205.243.166  github.com.localdomain
```

Test from a netshoot pod within argocd NS

```
#  k run netshoot -n argocd --rm -it --image=nicolaka/netshoot -- bash
If you don't see a command prompt, try pressing enter.
netshoot:~# curl -I https://github.com
HTTP/2 200
date: Wed, 24 Sep 2025 12:36:54 GMT
content-type: text/html; charset=utf-8
vary: X-PJAX, X-PJAX-Container, Turbo-Visit, Turbo-Frame, X-Requested-With, Accept-
Language,Accept-Encoding, Accept, X-Requested-With
content-language: en-US
etag: W/"bd1f03eeece7fbf502d53ea53e83bbee"
cache-control: max-age=0, private, must-revalidate
```

strict-transport-security: max-age=31536000; includeSubdomains; preload
x-frame-options: deny
x-content-type-options: nosniff
x-xss-protection: 0
referrer-policy: origin-when-cross-origin, strict-origin-when-cross-origin
content-security-policy: default-src 'none'; base-uri 'self'; child-src github.githubassets.com github.com/assets-cdn/worker/ github.com/assets/ gist.github.com/assets-cdn/worker/; connect-src 'self' uploads.github.com www.githubstatus.com collector.github.com raw.githubusercontent.com api.github.com github-cloud.s3.amazonaws.com github-production-repository-file-5c1aeb.s3.amazonaws.com github-production-upload-manifest-file-7fdce7.s3.amazonaws.com github-production-user-asset-6210df.s3.amazonaws.com *.rel.tunnels.api.visualstudio.com wss://*.rel.tunnels.api.visualstudio.com github.githubassets.com objects-origin.githubusercontent.com copilot-proxy.githubusercontent.com proxy.individual.githubcopilot.com proxy.business.githubcopilot.com proxy.enterprise.githubcopilot.com *.actions.githubusercontent.com wss://*.actions.githubusercontent.com productionresultssa0.blob.core.windows.net/ productionresultssa1.blob.core.windows.net/ productionresultssa2.blob.core.windows.net/ productionresultssa3.blob.core.windows.net/ productionresultssa4.blob.core.windows.net/ productionresultssa5.blob.core.windows.net/ productionresultssa6.blob.core.windows.net/ productionresultssa7.blob.core.windows.net/ productionresultssa8.blob.core.windows.net/ productionresultssa9.blob.core.windows.net/ productionresultssa10.blob.core.windows.net/ productionresultssa11.blob.core.windows.net/ productionresultssa12.blob.core.windows.net/ productionresultssa13.blob.core.windows.net/ productionresultssa14.blob.core.windows.net/ productionresultssa15.blob.core.windows.net/ productionresultssa16.blob.core.windows.net/ productionresultssa17.blob.core.windows.net/ productionresultssa18.blob.core.windows.net/ productionresultssa19.blob.core.windows.net/ github-production-repository-image-32fea6.s3.amazonaws.com github-production-release-asset-2e65be.s3.amazonaws.com insights.github.com wss://alive.github.com wss://alive-staging.github.com api.githubcopilot.com api.individual.githubcopilot.com api.business.githubcopilot.com api.enterprise.githubcopilot.com edge.fullstory.com rs.fullstory.com; font-src github.githubassets.com; form-action 'self' github.com gist.github.com copilot-workspace.githubnext.com objects-origin.githubusercontent.com; frame-ancestors 'none'; frame-src viewscreen.githubusercontent.com notebooks.githubusercontent.com www.youtube-nocookie.com; img-src 'self' data: blob: github.githubassets.com media.githubusercontent.com camo.githubusercontent.com identicons.github.com avatars.githubusercontent.com private-avatars.githubusercontent.com github-cloud.s3.amazonaws.com objects.githubusercontent.com release-assets.githubusercontent.com secured-user-images.githubusercontent.com/ user-images.githubusercontent.com/ private-user-images.githubusercontent.com opengraph.githubassets.com marketplace-screenshots.githubusercontent.com/ copilotprodattachments.blob.core.windows.net/github-production-copilot-attachments/ github-production-user-asset-6210df.s3.amazonaws.com customer-stories-feed.github.com spotlights-feed.github.com objects-origin.githubusercontent.com *.githubusercontent.com images.ctfassets.net/8aevphvgewt8/; manifest-src 'self'; media-src github.com user-images.githubusercontent.com/ secured-user-images.githubusercontent.com/ private-user-images.githubusercontent.com github-production-user-asset-6210df.s3.amazonaws.com gist.github.com github.githubassets.com assets.ctfassets.net/8aevphvgewt8/ videos.ctfassets.net/8aevphvgewt8/; script-src github.githubassets.com; style-src 'unsafe-inline' github.githubassets.com; upgrade-insecure-requests; worker-src github.githubassets.com github.com/assets-cdn/worker/ github.com/assets/ gist.github.com/assets-cdn/worker/
server: github.com
accept-ranges: bytes
set-cookie: _gh_sess=TwwwcmStrg21ZdQNQmGKXhYqf%2Fi%2BbWGEP%2Bo%2B%

```
2F4w7vkmUVWTu%2FgNInziOsQzqwuDgJH0aUxrkpa4iD7xgDEvX%2FsAMXmC%2FyXVzuu%
2BG6t1hO6mlXow%2BIN7ou4qNdAEMnrrWSYUUvrW59hzHb2H3%
2FHkbpFmbUZZ6EmZZIr8tHMCg72Ell7U3e8TtzSzA0nLxJNHLHV4IXPUc%2BqI4EJT9%
2F11fMtKvOW7e5%2BhBwqYzQH%
2FPWl8sLiFm6TIEQy0TgmtJOGfnTF913Bp3ce9Geby31sm8AA%3D%3D--CR7j3l4eHhdkAD8%2B--
UI3I5S3DWRszU72mIAEJsA%3D%3D; Path=/; HttpOnly; Secure; SameSite=Lax
set-cookie: _octo=GH1.1.613555136.1758717416; Path=/; Domain=github.com; Expires=Thu, 24
Sep 2026 12:36:56 GMT; Secure; SameSite=Lax
set-cookie: logged_in=no; Path=/; Domain=github.com; Expires=Thu, 24 Sep 2026 12:36:56 GMT;
HttpOnly; Secure; SameSite=Lax
x-github-request-id: 9C9D:2CCE73:1BC0738:20A6B22:68D3E5E7

netshoot:~# ping -c 4 github.com
PING github.com (20.205.243.166) 56(84) bytes of data.
64 bytes from 20.205.243.166: icmp_seq=1 ttl=127 time=105 ms
64 bytes from 20.205.243.166: icmp_seq=2 ttl=127 time=330 ms
64 bytes from 20.205.243.166: icmp_seq=3 ttl=127 time=44.9 ms
^C
--- github.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 44.913/159.731/329.677/122.615 ms
```

I adjusted git path and branch name and git URL.

Still Unknown sync status.
When I ran below:

```
# k logs -n argocd argocd-application-controller-0

{"level":"warning","msg":"Failed to save cluster info: dial tcp 10.43.186.128:6379: i/o
timeout","time":"2025-09-24T12:20:00Z"}
{"level":"warning","msg":"Failed to save cluster info: dial tcp 10.43.186.128:6379: i/o
timeout","time":"2025-09-24T12:20:15Z"}
{"level":"warning","msg":"Failed to save cluster info: dial tcp 10.43.186.128:6379: i/o
timeout","time":"2025-09-24T12:20:25Z"}
```

6379 is redis bro. There shld be an argocd redis pod which application controller uses to persist or fetch state. If not present, it will be an endless Unknown sync status.

Below we see there is a argocd-redis pod and deploy

```
# k get po -n argocd
NAME                                  READY   STATUS    RESTARTS      AGE
argocd-application-controller-0       1/1     Running   1 (47m ago)   3d4h
argocd-applicationset-controller-54f96997f8-m2bjb   1/1     Running   1 (47m ago)   3d4h
argocd-dex-server-798cbff4c7-pg447    1/1     Running   1 (47m ago)   3d4h
argocd-notifications-controller-644f66f7df-qw6tc    1/1     Running   1 (47m ago)   3d4h
argocd-redis-6684c6947f-2lkwq         1/1     Running   1 (47m ago)   3d4h
argocd-repo-server-8664486b9d-zknpr   1/1     Running   0             13m
argocd-server-64d5fcbd58-8qbdt        1/1     Running   1 (47m ago)   3d4h

# k get svc -n argocd
NAME                              TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)              AGE
argocd-applicationset-controller  ClusterIP   10.43.125.26   <none>        7000/TCP,8080/TCP
3d4h
```

```
argocd-dex-server                   ClusterIP   10.43.151.178   <none>
5556/TCP,5557/TCP,5558/TCP   3d4h
argocd-metrics                      ClusterIP   10.43.246.181   <none>      8082/TCP              3d4h
argocd-notifications-controller-metrics   ClusterIP   10.43.179.58   <none>      9001/TCP
3d4h
argocd-redis                        ClusterIP   10.43.186.128   <none>      6379/TCP              3d4h
argocd-repo-server                  ClusterIP   10.43.25.69    <none>      8081/TCP,8084/TCP
3d4h
argocd-server                       NodePort    10.43.157.58    <none>
80:31805/TCP,443:32619/TCP   3d4h
argocd-server-metrics               ClusterIP   10.43.22.170    <none>      8083/TCP
3d4h

# k get deploy,sts -n argocd | grep redis
deployment.apps/argocd-redis                  1/1    1          1          3d4h
```

I checked logs of app

```
# k describe app -n argocd sampapp

Status:
  Conditions:
    Last Transition Time:  2025-09-24T12:48:16Z
    Message:               Failed to load target state: failed to generate manifest for source 1 of 1: rpc
error: code = Unknown desc = failed to list refs: dial tcp 10.43.186.128:6379: i/o timeout
    Type:                  ComparisonError
```

Checked if endpoints of svc matches pod and if redis pod listening on 6379

```
# k exec -n argocd -it argocd-redis-6684c6947f-2lkwq -- netstat -tlnp | grep 6379
Defaulted container "redis" out of: redis, secret-init (init)
tcp        0      0 0.0.0.0:6379            0.0.0.0:*               LISTEN      1/redis-server *:63
tcp        0      0 :::6379                 :::*                    LISTEN      1/redis-server *:63

$ k -n argocd get endpoints argocd-redis
Warning: v1 Endpoints is deprecated in v1.33+; use discovery.k8s.io/v1 EndpointSlice
NAME            ENDPOINTS             AGE
argocd-redis    172.18.137.119:6379   3d4h

$ k get po -n argocd -owide | grep argocd-redis -i
argocd-redis-6684c6947f-2lkwq                 1/1    Running   1 (52m ago)   3d4h   172.18.137.119
master1   <none>          <none>
```

Tried to apply netpol to allow connection

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-argocd-controller-to-redis
  namespace: argocd
spec:
  podSelector:
    matchLabels:
      app.kubernetes.io/name: argocd-redis
```

```
  ingress:
    - from:
        - podSelector:
            matchLabels:
              app.kubernetes.io/name: argocd-application-controller
      ports:
        - protocol: TCP
          port: 6379
  policyTypes:
  - Ingress
```

Reapplied argocd manifest

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

Restart controller and repo server

```
# kubectl rollout restart deployment -n argocd argocd-repo-server
# kubectl rollout restart statefulset -n argocd argocd-application-controller
```

Status still Unknown for sync status. Health status Healthy

Checked connectivity to redis pod as above logs always complained about redis connectivity

```
# k run netshoot --rm -it --image=nicolaka/netshoot -n argocd -- sh
If you don't see a command prompt, try pressing enter.
~ # nc -zv argocd-redis 6379
```

It hung

Edited the netpol from above to include all allowed in argocd ns

```
$ k get netpol -n argocd allow-argocd-all-to-redis -oyaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |

{"apiVersion":"networking.k8s.io/v1","kind":"NetworkPolicy","metadata":{"annotations":{},"name":"allow-argocd-all-to-
redis","namespace":"argocd"},"spec":{"ingress":[{"from":[{"namespaceSelector":{"matchLabels":{"kubernetes.io/metadata.name":"argocd"}}}],"ports":[{"port":6379,"protocol":"TCP"}]}],"podSelector":{"matchLabels":{"app.kubernetes.io/name":"argocd-redis"}},"policyTypes":["Ingress"]}}
  creationTimestamp: "2025-09-25T06:21:08Z"
  generation: 1
  name: allow-argocd-all-to-redis
  namespace: argocd
  resourceVersion: "102689"
  uid: a858e622-2006-4051-a651-80095396e8b6
spec:
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
```

```
      kubernetes.io/metadata.name: argocd
  ports:
  - port: 6379
    protocol: TCP
 podSelector:
  matchLabels:
   app.kubernetes.io/name: argocd-redis
 policyTypes:
 - Ingress
```

Tried to test connectivity to redis pod with netshoot pod again but still hung.

Found this netpol:

```
$ k get netpol -n argocd argocd-redis-network-policy -oyaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
 annotations:
  kubectl.kubernetes.io/last-applied-configuration: |

{"apiVersion":"networking.k8s.io/v1","kind":"NetworkPolicy","metadata":{"annotations":{},"labels":{"app.kubernetes.io/component":"redis","app.kubernetes.io/name":"argocd-redis","app.kubernetes.io/part-of":"argocd"},"name":"argocd-redis-network-policy","namespace":"argocd"},"spec":{"ingress":[{"from":[{"podSelector":{"matchLabels":{"app.kubernetes.io/name":"argocd-server"}}},{"podSelector":{"matchLabels":{"app.kubernetes.io/name":"argocd-repo-server"}}},{"podSelector":{"matchLabels":{"app.kubernetes.io/name":"argocd-application-controller"}}}],"ports":[{"port":6379,"protocol":"TCP"}]}],"podSelector":{"matchLabels":{"app.kubernetes.io/name":"argocd-redis"}},"policyTypes":["Ingress"]}}
 creationTimestamp: "2025-09-21T08:19:13Z"
 generation: 1
 labels:
  app.kubernetes.io/component: redis
  app.kubernetes.io/name: argocd-redis
  app.kubernetes.io/part-of: argocd
 name: argocd-redis-network-policy
 namespace: argocd
 resourceVersion: "94067"
 uid: f7560754-bf94-460a-82d4-42a149122597
spec:
 ingress:
 - from:
  - podSelector:
     matchLabels:
      app.kubernetes.io/name: argocd-server
  - podSelector:
     matchLabels:
      app.kubernetes.io/name: argocd-repo-server
  - podSelector:
     matchLabels:
      app.kubernetes.io/name: argocd-application-controller
  ports:
  - port: 6379
```

```
    protocol: TCP
  podSelector:
    matchLabels:
      app.kubernetes.io/name: argocd-redis
  policyTypes:
  - Ingress
```

No wonder it hung - it only explicitly allows traffic from argocd-server and repo server and app controller.

But after applying the all to redis netpol, now the argocd-repo-server does not complain to timeout errors to redis IP.

Now I check the application sampapp it is still Unknown, with below error:

```
Status:
  Conditions:
    Last Transition Time:  2025-09-25T06:09:24Z
    Message:               Failed to load target state: failed to generate manifest for source 1 of 1: rpc error: code = Unavailable desc = connection error: desc = "transport: Error while dialing: dial tcp 10.43.25.69:8081: i/o timeout
```

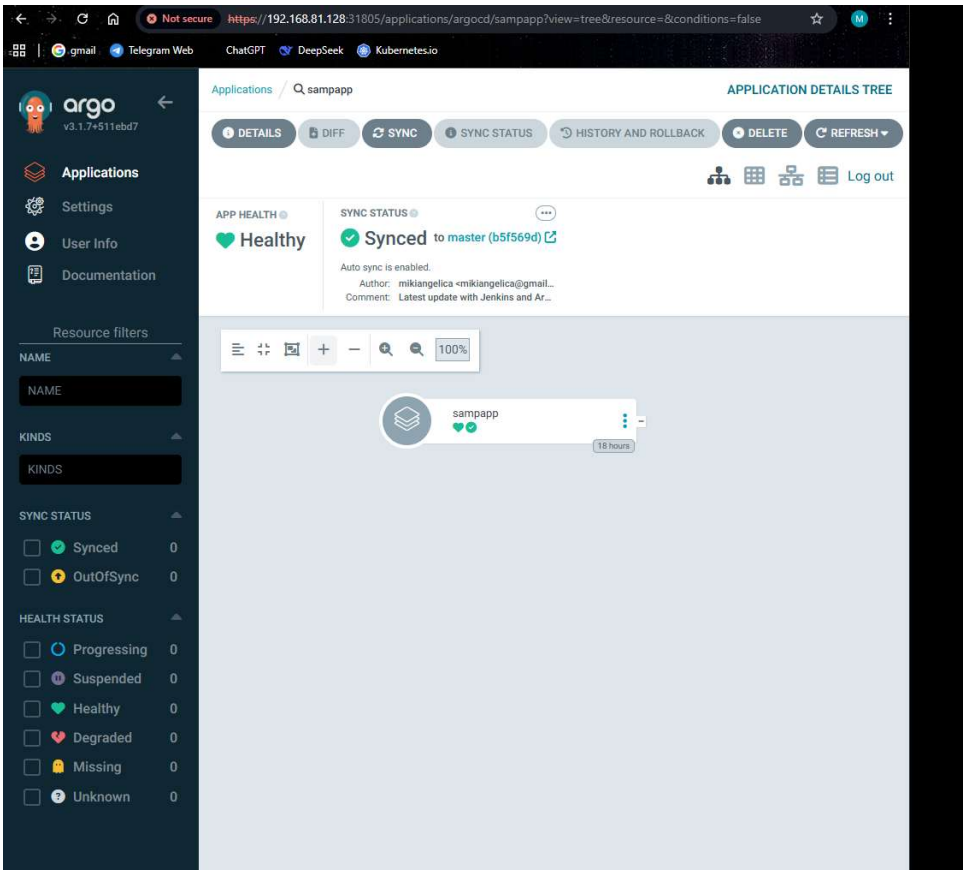
Previously, on ArgocdUI it said 2 errors. Now after fixing the redis thing it is 1 error.

The above log states the argoCD app controller cannot reach the repo server on port 8081

Got some issues with the netpols!!! There are multiple default netpols (when applying the origin argocd manifest)
I deleted ALL netpols in argocd - will reapply some later on.

NOW my cluster sync status is Synced.

```
# k get app sampapp -n argocd
NAME      SYNC STATUS   HEALTH STATUS
sampapp   Synced        Healthy
```



ISSUE 4: BUILD FAILED 1
Current deployment.yaml and image tag in DockerHub are same - v1
Changed deployment.yaml's image tag to oldtag instead of v1

Triggered Build and it failed because current deployment.yaml and the image tag in DockerHub are the same - v1
Changed deployment.yaml's image tag to oldtag instead of v1.

Triggered build again.

ISSUE 5: BUILD FAILED 2
Indentation error in Jenkinsfile
Added 4 tabs in Jenkinsfile before image to handle indentation

Build failed again

#2 (Sep 25, 2025, 10:01:32 AM) Add description Keep this build forever Started 2 min 11 sec ago Took 25 sec Started by user Jenkins Admin This run spent: 12 sec waiting; 25 sec build duration; 25 sec total from scheduled to completion. Revision: f8bf15ec15040f22b06a4f90a63c59253e509269 Repository: https://github.com/mikiangelica/sampapp.git refs/remotes/origin/master Revision: f8bf15ec15040f22b06a4f90a63c59253e509269 Repository: https://github.com/mikiangelica/k8s-manifests.git refs/remotes/origin/master The following steps that have been detected may have insecure interpolation of sensitive variables (click here for an explanation): sh: [DOCKER_PASS] No changes.

Why?

The `sed -i 's|image: .*|image: $IMAGE_NAME:$IMAGE_TAG|'` `manifests/base/deployment.yaml` assumes your manifest has a line like:

yaml Copy code

```yaml
image: something
```

But in real-world YAMLs, it's usually indented:

yaml Copy code

```yaml
        image: mikiangelica/sampapp:oldtag
```

Because of the spaces, `sed` didn't match it.

Edited Jenkinsfile to have 4 tabs before image so it can handle indentation:

```
sed -i 's|\\s*image: .*|        image: $IMAGE_NAME:$IMAGE_TAG|' manifests/base/deployment.yaml
```

ISSUE 6: BUILD FAILED 3

Docker not found

Made jenkins-agent file for jenkins to use

Failed again. Checked pipeline logs and saw that pipeline failed at this line

```
docker: not found
```

By default, Jenkins agent pod (jenkins/inbound-agent) does not include Docker. That's why can't build or push

Make Dockerfile.jenkins-agent file under Dockerfiles/jenkins/

```
FROM jenkins/inbound-agent:latest

USER root

# Install Docker CLI and buildx
RUN apt-get update && \
    apt-get install -y curl apt-transport-https ca-certificates gnupg lsb-release && \
    curl -fsSL https://download.docker.com/linux/debian/gpg | apt-key add - && \
    echo "deb [arch=amd64] https://download.docker.com/linux/debian $(lsb_release -cs) stable" \
      > /etc/apt/sources.list.d/docker.list && \
    apt-get update && \
    apt-get install -y docker-ce-cli && \
    mkdir -p ~/.docker/cli-plugins && \
    curl -L https://github.com/docker/buildx/releases/download/v0.16.2/buildx-v0.16.2.linux-amd64 \
      -o ~/.docker/cli-plugins/docker-buildx && \
    chmod +x ~/.docker/cli-plugins/docker-buildx

USER admin
```

```
# docker login
# docker buildx build -t mikiangelica/jenkins-agent-buildx:latest -f Dockerfile.jenkins-agent .
```

```
# docker push mikiangelica/jenkins-agent-buildx:latest
```

Registering custom agent in Jenkins
Add Globally in Jenkins UI
Go to Jenkins > Manage Jenkins > Clouds > Kubernetes
Under Pod Templates, add new template:

- Name: docker-build-agent
- Namespace: jenkins
- Labels: docker-build
- Usage: Only build jobs with label expressions matching this node
- Pod tempate to inherit from: (empty)
- Name of the container that will run the Jenkins agent: jnlp
- ☑ "Inject Jenkins agent in agent container"
- Containers
    - Name: jnlp
    - Docker image: mikiangelica/jenkins-agent-buildx:latest
    - ☑ "Always pull image"
    - Working dir: /home/jenkins/agent
    - Command to run: sleep
    - Args: 9999999
- Service Account: jenkins
- Workspace Volume: Empty Dir Workspace Volume

Click Create button
This will register the pod template globally so that any pipeline can use the label docker-build given.

Edit Jenkinsfile to accept the jenkins agent
Original:

```
pipeline {
  agent any
  environment {
    IMAGE_NAME = "mikiangelica/sampapp"
    IMAGE_TAG  = "v1"
  }
```

Adjusted:

```
pipeline {
  agent { label 'docker-build' }
  environment {
    IMAGE_NAME = "mikiangelica/sampapp"
    IMAGE_TAG  = "v1"
  }
```

ISSUE 7: BUILD FAILED 4
Rerun pipeline… failed. I realise I forgot to push Jenkinsfile to GitHub. Pushed to GitHub and went for round 5 of pipeline run.

ISSUE 8: BUILD FAILED 5
This run FAILED again!!
Docker-build-agent pod could not find user
Amended user in Dockerfile.jenkins-agent
This time with different error… The docker-build-agent pod is coming up but with below error:
jenkins docker-build-agent-xrxpd 0/1 CreateContainerError 0 2m6s

```
Events:
  Type     Reason     Age              From               Message
  ----     ------     ----             ----               -------
  Normal   Scheduled  2m17s            default-scheduler  Successfully assigned jenkins/docker-build-agent-xrxpd to master1
  Normal   Pulling    2m17s            kubelet            Pulling image "jenkins/inbound-agent:3327.v868139a_d00e0-6"
  Normal   Pulled     87s              kubelet            Successfully pulled image "jenkins/inbound-agent:3327.v868139a_d00e0-6" in 49.376s (49.376s including waiting). Image size: 136833139 bytes.
  Normal   Created    87s              kubelet            Created container: set-up-jenkins-agent
  Normal   Started    87s              kubelet            Started container set-up-jenkins-agent
  Normal   Pulled     41s              kubelet            Successfully pulled image "mikiangelica/jenkins-agent-buildx:latest" in 46.266s (46.266s including waiting). Image size: 239373498 bytes.
  Warning  Failed     41s              kubelet            Error: failed to create containerd container: mount callback failed on /var/lib/rancher/k3s/agent/containerd/tmpmounts/containerd-mount126234113: no users found
  Normal   Pulled     37s              kubelet            Successfully pulled image "mikiangelica/jenkins-agent-buildx:latest" in 2.549s (2.549s including waiting). Image size: 239373498 bytes.
  Warning  Failed     37s              kubelet            Error: failed to create containerd container: mount callback failed on /var/lib/rancher/k3s/agent/containerd/tmpmounts/containerd-mount1069881075: no users found
  Normal   Pulled     20s              kubelet            Successfully pulled image "mikiangelica/jenkins-agent-buildx:latest" in 3.061s (3.061s including waiting). Image size: 239373498 bytes.
  Warning  Failed     20s              kubelet            Error: failed to create containerd container: mount callback failed on /var/lib/rancher/k3s/agent/containerd/tmpmounts/containerd-mount3507521232: no users found
  Normal   Pulling    9s (x4 over 87s)  kubelet           Pulling image "mikiangelica/jenkins-agent-buildx:latest"
  Normal   Pulled     6s               kubelet            Successfully pulled image "mikiangelica/jenkins-agent-buildx:latest" in 2.734s (2.734s including waiting). Image size: 239373498 bytes.
  Warning  Failed     6s               kubelet            Error: failed to create containerd container: mount callback failed on /var/lib/rancher/k3s/agent/containerd/tmpmounts/containerd-mount1962973324: no users found
```

Seeing this error:

```
Error: failed to create containerd container: mount callback failed on /var/lib/rancher/k3s/agent/containerd/tmpmounts/containerd-mount1962973324: no users found
```

Need to amend user in Dockerfile.jenkins-agent
Original:

```
FROM jenkins/inbound-agent:latest

USER root

# Install Docker CLI and buildx
RUN apt-get update && \
```

```
  apt-get install -y curl apt-transport-https ca-certificates gnupg lsb-release && \
  curl -fsSL https://download.docker.com/linux/debian/gpg | apt-key add - && \
  echo "deb [arch=amd64] https://download.docker.com/linux/debian $(lsb_release -cs) stable" \
   > /etc/apt/sources.list.d/docker.list && \
  apt-get update && \
  apt-get install -y docker-ce-cli && \
  mkdir -p ~/.docker/cli-plugins && \
  curl -L https://github.com/docker/buildx/releases/download/v0.16.2/buildx-v0.16.2.linux-amd64 \
   -o ~/.docker/cli-plugins/docker-buildx && \
  chmod +x ~/.docker/cli-plugins/docker-buildx

USER admin
```

Amended to:

```
FROM jenkins/inbound-agent:latest

USER root

# Install Docker CLI and buildx
RUN apt-get update && \
  apt-get install -y curl apt-transport-https ca-certificates gnupg lsb-release && \
  curl -fsSL https://download.docker.com/linux/debian/gpg | apt-key add - && \
  echo "deb [arch=amd64] https://download.docker.com/linux/debian $(lsb_release -cs) stable" \
   > /etc/apt/sources.list.d/docker.list && \
  apt-get update && \
  apt-get install -y docker-ce-cli && \
  mkdir -p ~/.docker/cli-plugins && \
  curl -L https://github.com/docker/buildx/releases/download/v0.16.2/buildx-v0.16.2.linux-amd64 \
   -o ~/.docker/cli-plugins/docker-buildx && \
  chmod +x ~/.docker/cli-plugins/docker-buildx

USER jenkins
```

The repushed Dockerfile to DockerHub:

```
# docker login
# docker buildx build -t mikiangelica/jenkins-agent-buildx:latest -f Dockerfile.jenkins-agent .
# docker push mikiangelica/jenkins-agent-buildx:latest
```

Deleted the docker-build-agent pod and reran pipeline…

ISSUE 9: BUILD FAILED 6
FAILED AGAIN
No Docker daemon running inside the Jenkins agent container
Added hostpath volume to jenkins pod template for jenkins agent (hosts' docker daemon socket file)

```
ERROR: Cannot connect to the Docker daemon at unix:///var/run/docker.sock. Is the docker daemon running?
```

So the Jenkins agent pod is now running with the custom image but the Jenkins agent container doesn't have a Docker daemon running.

We shall do this dirtily because we want to just test the pipeline. Edit the docker-build-agent pod

template under Jenkins > Clouds > kubernetes > pod template

So on the host machine (master node), the Docker daemon (dockerd) is running in the background…
That daemon exposes a **Unix socket file** at /var/run/docker.sock.

- This is a doorway into the Docker engine basically
- Any process that can read/write to this socket can issue Docker commands (build, push, run, remove containers etc)

Without the above mount, the Jenkins agent container has the docker CLI installed but no Docker daemon running inside… so when we run docker buildx build …, the CLI looks for the socket file at /var/run/docker.sock.
Inside the pod itself, the path doesn't exist (or is just an empty file so we get the above error. (cannot connect to Docker daemon at /var/run/docker.sock)

So with the mount, which in YAML looks like:

```
volumes:
- hostPath:
    path: /var/run/docker.sock
  name: docker-sock
containers:
- name: jnlp
  volumeMounts:
  - mountPath: /var/run/docker.sock
    name: docker-sock
```

It mounts the hosts' socket file /var/run/docker.sock into the container at the same path,, so that when the docker CLI inside the Jenkins agent tries to talk to /var/run/docker.sock - it is actually talking to the host's Docker daemon.

So the agent container provides the tools (docker, buildx etc)
Host machine does the real building, running, pushing
The container is just a remote control for Docker running outside.

Note: This setup works fine for testing, but in production it's risky (the agent gets root-level access to the host Docker). For prod, **Kaniko/BuildKit** can be used for a safer approach.
Security Implications here:

- Any process in the container can control all Docker containers on the host (root-level access)
- In shared clusters, this is considered insecure
- That's why in production, people use **Kaniko, BuildKit, or img** (which run builds without needing

the host Docker socket).

ISSUE 10: BUILD FAILED 7
Permission denied error
Removed USER jenkins line from Dockerfile.jenkins-agent-build
Reran pipeline after adding the hostpath volume above and it failed again. With permission denied error:

```
ERROR: permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Head "http://%2Fvar%2Frun%2Fdocker.sock/_ping": dial unix /var/run/docker.sock: connect: permission denied
```

So it seems we have mounted it correctly but permission denied.

Usually that socket file is owned by root and group docker on the host.
But in our container, Jenkins run as user jenkins (not root, and not in docker group)
So when docker CLI tries to talk to the socket, the kernel blocks it > permission denied.

Ok so now lets just adjust our Dockerfile so it stays as root. Previously we amended admin to jenkins. Now we just completely remove it

```
FROM jenkins/inbound-agent:latest

USER root

# Install Docker CLI and buildx
RUN apt-get update && \
   apt-get install -y curl apt-transport-https ca-certificates gnupg lsb-release && \
   curl -fsSL https://download.docker.com/linux/debian/gpg | apt-key add - && \
   echo "deb [arch=amd64] https://download.docker.com/linux/debian $(lsb_release -cs) stable" \
    > /etc/apt/sources.list.d/docker.list && \
   apt-get update && \
   apt-get install -y docker-ce-cli && \
   mkdir -p ~/.docker/cli-plugins && \
   curl -L https://github.com/docker/buildx/releases/download/v0.16.2/buildx-v0.16.2.linux-amd64 \
    -o ~/.docker/cli-plugins/docker-buildx && \
   chmod +x ~/.docker/cli-plugins/docker-buildx
```

Push to DockerHub again

```
# docker login
# docker buildx build -t mikiangelica/jenkins-agent-buildx:latest -f Dockerfile.jenkins-agent .
# docker push mikiangelica/jenkins-agent-buildx:latest
```

ISSUE 11: BUILD FAILED 8
Failed to read Dockerfile
Amended path to my sampapp Dockerfile

RUN PIPELINE ATTEMPT #8
Which FAILED again… lol

```
ERROR: failed to solve: failed to read dockerfile: open Dockerfile: no such file or directory
```

So this means we got past the docker not found & permish denied errors. But now it is looking for a file named Dockerfile in the workspace root… So we need to recheck where our sampapp Dockerfile is..

```
# /c/Users/mikia/OneDrive/Documents/SampApp/sampapp/Dockerfiles/sampapp
```

And in my Jenkins file it is

```
    stage('Build & Push Docker Image') {
      steps {
        withCredentials([usernamePassword(credentialsId: 'dockerhub-creds',
                                          usernameVariable: 'DOCKER_USER',
                                          passwordVariable: 'DOCKER_PASS')]) {
          sh """
            docker build -t $IMAGE_NAME:$IMAGE_TAG -f Dockerfiles/Dockerfile .
            echo $DOCKER_PASS | docker login -u $DOCKER_USER --password-stdin
            docker push $IMAGE_NAME:$IMAGE_TAG
          """
        }
      }
    }
```

Fixed to:

```
    stage('Build & Push Docker Image') {
      steps {
        withCredentials([usernamePassword(credentialsId: 'dockerhub-creds',
                                          usernameVariable: 'DOCKER_USER',
                                          passwordVariable: 'DOCKER_PASS')]) {
          sh """
            docker build -t $IMAGE_NAME:$IMAGE_TAG -f Dockerfiles/sampapp/Dockerfile .
            echo $DOCKER_PASS | docker login -u $DOCKER_USER --password-stdin
            docker push $IMAGE_NAME:$IMAGE_TAG
          """
        }
      }
    }
```

ISSUE 12: BUILD FAILED 9
No github credentials in Jenkinsfile
Added github credentials in Jenkinsfile (github PAT)

RUN PIPELINE ATTEMPT #9
Failed due to this error:

```
+ sed -i s|image: .*|          image: mikiangelica/sampapp:v1| manifests/base/deployment.yaml
+ git config user.email mikiangelica@gmail.com
+ git config user.name admin
+ git add manifests/base/deployment.yaml
+ git commit -m Update image to mikiangelica/sampapp:v1
[master 8ee8443] Update image to mikiangelica/sampapp:v1
 1 file changed, 1 insertion(+), 1 deletion(-)
+ git push origin master
fatal: could not read Username for 'https://github.com': No such device or address
```

BTW our image built and pushed successfully but the error is about pushing the updated manifest back to Github Repo.

This happens because:

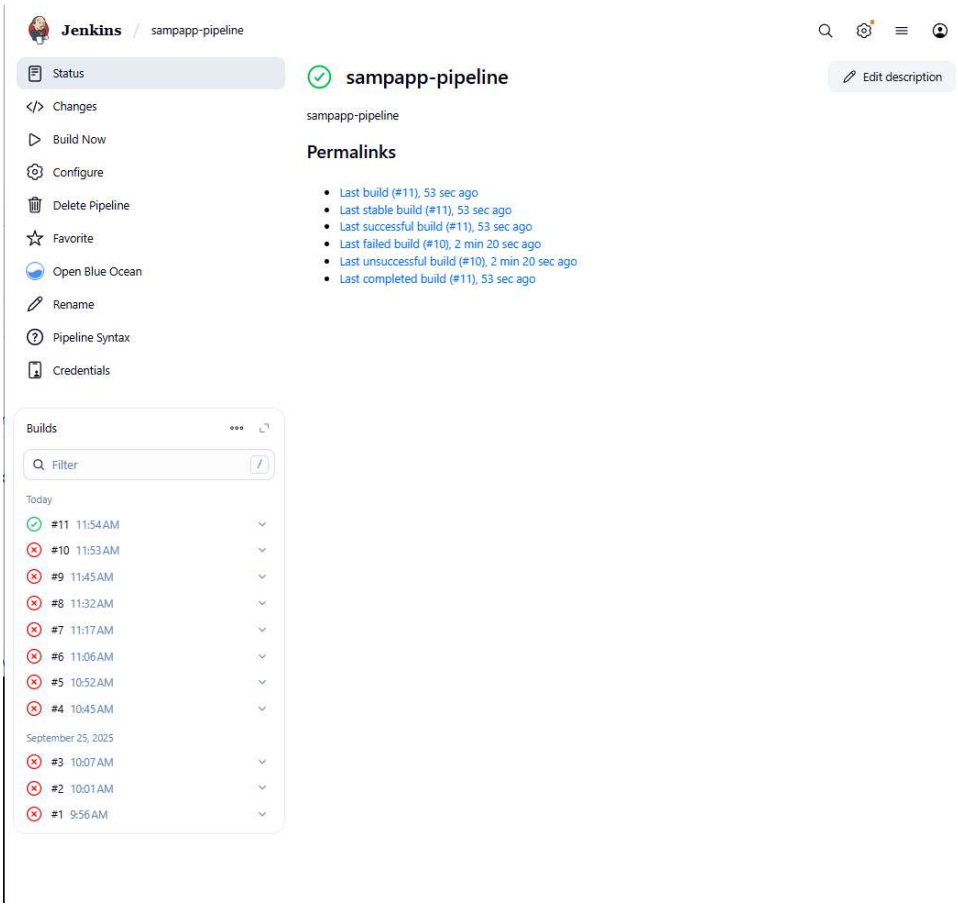
- Jenkins is running inside your custom agent pod.
- That pod doesn’t have your GitHub credentials configured globally.
- So when git push runs, it can’t authenticate to GitHub.

Need to add withCredentials block to Jenkinfile:

```
    stage('Update Manifests') {
      steps {
        withCredentials([usernamePassword(credentialsId: 'github-pat',
                                          usernameVariable: 'GIT_USER',
                                          passwordVariable: 'GIT_PASS')]) {
        sh """
          sed -i 's|image: .*|      image: $IMAGE_NAME:$IMAGE_TAG|'
manifests/base/deployment.yaml
          git config user.email "mikiangelica@gmail.com"
          git config user.name "admin"
          git add manifests/base/deployment.yaml
          git commit -m "Update image to $IMAGE_NAME:$IMAGE_TAG" || echo "No changes to
commit"
          git push https://$GIT_USER:$GIT_PASS@github.com/mikiangelica/k8s-manifests.git master
        """
        }
      }
    }
```

Repushed Jenkinsfile to Github. Had to push twice more (because 10th try I missed out 1 } in the Jenkinsfile which caused it to fail)

Finally 11th time's the charm! It succeeded….


Jenkins / sampapp-pipeline
Status
Changes
Build Now
Configure
Delete Pipeline
Favorite
Open Blue Ocean
Rename
Pipeline Syntax
Credentials
Builds
Filter
Today
#11 11:54 AM
#10 11:53 AM
#9 11:45 AM
#8 11:32 AM
#7 11:17 AM
#6 11:06 AM
#5 10:52 AM
#4 10:45 AM
September 25, 2025
#3 10:07 AM
#2 10:01 AM
#1 9:56 AM
sampapp-pipeline
Edit description
sampapp-pipeline
Permalinks
Last build (#11), 53 sec ago
Last stable build (#11), 53 sec ago
Last successful build (#11), 53 sec ago
Last failed build (#10), 2 min 20 sec ago
Last unsuccessful build (#10), 2 min 20 sec ago
Last completed build (#11), 53 sec ago