

Configuration Manual

MSc Research Project
MSc In Cloud Computing

Prasanna Shinde
Student ID: X23278480

School of Computing
National College of Ireland

Supervisor: Sai Emani

National College of Ireland
MSc Project Submission Sheet
School of Computing



Student Name: Prasanna Shinde
Student ID: X23278480
Programme: MSc in Cloud Computing
Year: 2024
Module: MSc Research Project
Lecturer: Sai Emani
Submission Due Date: 11th August 2025
Project Title: Self-Healing CI/CD Pipelines: Automating Kubernetes Deployments with Terraform and Ansible.

Word Count:1468 **Page Count:** 20

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Prasanna Shinde
Date: 10th August 2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	<input type="checkbox"/>
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	<input type="checkbox"/>
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	<input type="checkbox"/>

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Prasanna Shinde
Student ID: X23278480

1 Introduction

This configuration guide is a point-by-point tutorial on architecting and deploying a self-healing CI/CD pipeline based on infrastructure provisioning by Terraform, a service layer by Flask web application and PostgreSQL, CI/CD orchestration by Jenkins, monitoring by Prometheus and policy enforcement by Open Policy Agent (OPA) Gatekeeper. The custom Python modules used in the system includes failure classification, trust-aware deployment, dynamic policy deployment, deployment health scoring and explainable rollbacks. It addresses failure of deployment, violations of policies, crashes of applications, and automatically recovers based on predefined workflows that are initiated by Ansible and Kubernetes. Each configuration is Infrastructure as Code (IaC) based to provide repeatability, scalability, and compliance of each deployment.

2 System Requirements

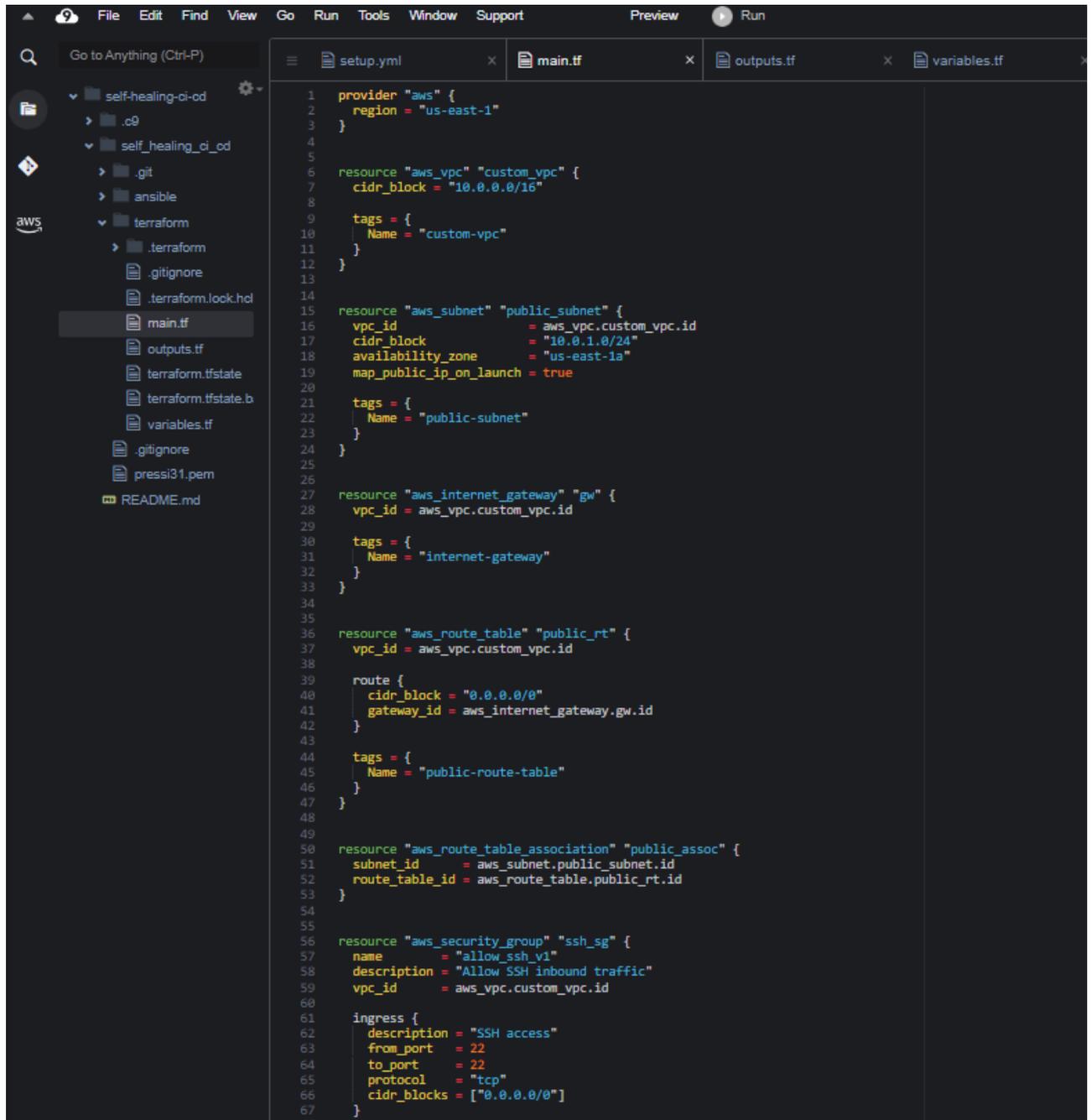
This section provides the minimum hardware, software, and network conditions to be used to install and operate the experimental environment of the Self-Healing CI/CD Pipelines: Automating Kubernetes Deployments with Terraform and Ansible.

- Operating System: Ubuntu 22.04 LTS
- Terraform: v1.7+
- Ansible: v2.15
- Jenkins: Latest
- Docker: v25+
- cri-dockerd: v0.3.4+
- Kubernetes: v1.30+
- Helm: v3.14+
- Prometheus: v2.53+
- OPA Gatekeeper: v3.16+
- Python: v3.10+
- Cloud Platform: AWS EC2 Instance

3 Setup Guide

3.1 Terraform

The Terraform configuration specifies details on the cloud infrastructure of the experiment. The main.tf file is used to provision the AWS EC2 instance with all resources needed, including security groups, IAM roles, and networking aspects of self-healing CI/CD pipeline.



```
1 provider "aws" {
2   region = "us-east-1"
3 }
4
5
6 resource "aws_vpc" "custom_vpc" {
7   cidr_block = "10.0.0.0/16"
8
9   tags = {
10    Name = "custom-vpc"
11  }
12 }
13
14
15 resource "aws_subnet" "public_subnet" {
16   vpc_id            = aws_vpc.custom_vpc.id
17   cidr_block        = "10.0.1.0/24"
18   availability_zone = "us-east-1a"
19   map_public_ip_on_launch = true
20
21   tags = {
22    Name = "public-subnet"
23  }
24 }
25
26
27 resource "aws_internet_gateway" "gw" {
28   vpc_id = aws_vpc.custom_vpc.id
29
30   tags = {
31    Name = "internet-gateway"
32  }
33 }
34
35
36 resource "aws_route_table" "public_rt" {
37   vpc_id = aws_vpc.custom_vpc.id
38
39   route {
40     cidr_block = "0.0.0.0/0"
41     gateway_id = aws_internet_gateway.gw.id
42   }
43
44   tags = {
45    Name = "public-route-table"
46  }
47 }
48
49
50 resource "aws_route_table_association" "public_assoc" {
51   subnet_id      = aws_subnet.public_subnet.id
52   route_table_id = aws_route_table.public_rt.id
53 }
54
55
56 resource "aws_security_group" "ssh_sg" {
57   name            = "allow_ssh_v1"
58   description     = "Allow SSH inbound traffic"
59   vpc_id          = aws_vpc.custom_vpc.id
60
61   ingress {
62     description = "SSH access"
63     from_port   = 22
64     to_port     = 22
65     protocol    = "tcp"
66     cidr_blocks = ["0.0.0.0/0"]
67   }
68 }
```

```

60 ingress {
61   description = "Jenkins Web UI"
62   from_port   = 8080
63   to_port     = 8080
64   protocol    = "tcp"
65   cidr_blocks = ["0.0.0.0/0"]
66 }
67
68 ingress {
69   description = "HTTP for NodePort services"
70   from_port   = 30080
71   to_port     = 32767
72   protocol    = "tcp"
73   cidr_blocks = ["0.0.0.0/0"]
74 }
75
76
77 ingress {
78   description = "Prometheus UI"
79   from_port   = 9090
80   to_port     = 9090
81   protocol    = "tcp"
82   cidr_blocks = ["0.0.0.0/0"]
83 }
84
85 ingress {
86   description = "OPA Gatekeeper webhook"
87   from_port   = 8443
88   to_port     = 8443
89   protocol    = "tcp"
90   cidr_blocks = ["0.0.0.0/0"]
91 }
92
93 ingress {
94   description = "Kubelet API"
95   from_port   = 10250
96   to_port     = 10250
97   protocol    = "tcp"
98   cidr_blocks = ["0.0.0.0/0"]
99 }
100
101 egress {
102   from_port = 0
103   to_port   = 0
104   protocol  = "-1"
105   cidr_blocks = ["0.0.0.0/0*"]
106 }
107
108 tags = {
109   Name = "ssh-access"
110 }
111
112 resource "aws_instance" "ci_cd_node" {
113   ami           = "ami-093b0d53c279acc90" # Ubuntu 22.04 LTS for us-east-1
114   instance_type = "t3.medium"
115   subnet_id     = aws_subnet.public_subnet.id
116   key_name      = var.key_name
117   vpc_security_group_ids = [aws_security_group.ssh_sg.id]
118   associate_public_ip_address = true
119
120   root_block_device {
121     volume_size = 50
122     volume_type = "gp3"
123     tags = {
124       Name = "root-volume"
125     }
126   }
127 }

```

```

106   cidr_blocks = ["0.0.0.0/0"]
107 }
108
109 egress {
110   from_port = 0
111   to_port   = 0
112   protocol  = "-1"
113   cidr_blocks = ["0.0.0.0/0"]
114 }
115
116 tags = {
117   Name = "ssh-access"
118 }
119
120 resource "aws_instance" "ci_cd_node" {
121   ami           = "ami-093b0d53c279acc90" # Ubuntu 22.04 LTS for us-east-1
122   instance_type = "t3.medium"
123   subnet_id     = aws_subnet.public_subnet.id
124   key_name      = var.key_name
125   vpc_security_group_ids = [aws_security_group.ssh_sg.id]
126   associate_public_ip_address = true
127
128   root_block_device {
129     volume_size = 50
130     volume_type = "gp3"
131     tags = {
132       Name = "root-volume"
133     }
134   }
135
136   tags = {
137     Name = "ci-cd-node"
138   }
139 }
140
141
142 resource "aws_ebs_volume" "data_disk" {
143   availability_zone = "us-east-1a"
144   size              = 100
145   type              = "gp3"
146
147   lifecycle {
148     create_before_destroy = true
149   }
150
151   tags = {
152     Name = "ci-cd-data"
153   }
154 }
155
156 resource "aws_volume_attachment" "data_attach" {
157   device_name = "/dev/sdf" # Will appear as /dev/xvdf
158   volume_id   = aws_ebs_volume.data_disk.id
159   instance_id = aws_instance.ci_cd_node.id
160   depends_on = [aws_instance.ci_cd_node]
161 }
162
163 resource "aws_eip" "ec2_ip" {
164   instance = aws_instance.ci_cd_node.id
165   depends_on = [aws_internet_gateway.gw]
166 }
167

```

These commands are used to initialize the infrastructure (Terraform)

- terraform init
- terraform apply -var="key_name=pressi31" -auto-approve

3.2 Ansible

Ansible script is use for installing the require tools in the system such as Docker, Kubernetes, OPA, Prometheus, Helm and Jenkins

```
setup.yml x main.tf x outputs.tf x variables.tf x
1 - name: Configure CI/CD node with Docker and cri-dockerd
2 hosts: ci_cd_node
3 become: true
4 tasks:
5
6   - name: Format /dev/nvme1n1 as ext4
7     filesystem:
8       fstype: ext4
9       dev: /dev/nvme1n1
10
11  - name: Create /mnt/data directory
12    file:
13      path: /mnt/data
14      state: directory
15      mode: '0755'
16
17  - name: Mount /dev/nvme1n1 to /mnt/data
18    mount:
19      path: /mnt/data
20      src: /dev/nvme1n1
21      fstype: ext4
22      opts: defaults,nofail
23      state: mounted
24
25  - name: Persist /mnt/data in fstab
26    lineinfile:
27      path: /etc/fstab
28      line: "/dev/nvme1n1 /mnt/data ext4 defaults,nofail 0 2"
29      create: yes
30
31  # Migrate Jenkins data to /mnt/data/jenkins
32  - name: Move Jenkins data to /mnt/data/jenkins
33    shell: |
34      systemctl stop jenkins || true
35      mkdir -p /mnt/data/jenkins
36      chown -R jenkins:jenkins /mnt/data/jenkins
37      mv /var/lib/jenkins/* /mnt/data/jenkins/
38      rm -rf /var/lib/jenkins
39      ln -s /mnt/data/jenkins /var/lib/jenkins
40    args:
41      creates: /mnt/data/jenkins
42
43  # Migrate Prometheus data to /mnt/data/prometheus
44  - name: Move Prometheus data to /mnt/data/prometheus
45    shell: |
46      systemctl stop prometheus || true
47      mkdir -p /mnt/data/prometheus
48      chown -R prometheus:prometheus /mnt/data/prometheus
49      mv /var/lib/prometheus/* /mnt/data/prometheus/
50      rm -rf /var/lib/prometheus
51      ln -s /mnt/data/prometheus /var/lib/prometheus
52    args:
53      creates: /mnt/data/prometheus
54
55  # Migrate Docker data to /mnt/data/docker
56  - name: Move Docker data to /mnt/data/docker
57    shell: |
58      systemctl stop docker || true
59      mkdir -p /mnt/data/docker
60      mv /var/lib/docker/* /mnt/data/docker/
61      rm -rf /var/lib/docker
62      ln -s /mnt/data/docker /var/lib/docker
63    args:
64      creates: /mnt/data/docker
65
```

```
66 # Migrate Grafana data to /mnt/data/grafana (optional)
67 - name: Move Grafana data to /mnt/data/grafana
68   shell: |
69     systemctl stop grafana-server || true
70     mkdir -p /mnt/data/grafana
71     chown -R grafana:grafana /mnt/data/grafana
72     mv /var/lib/grafana/* /mnt/data/grafana/
73     rm -rf /var/lib/grafana
74     ln -s /mnt/data/grafana /var/lib/grafana
75   args:
76     creates: /mnt/data/grafana
77
78 # Clean previous Jenkins/GPG configs
79 - name: Remove old Jenkins sources list
80   file:
81     path: /etc/apt/sources.list.d/jenkins.list
82     state: absent
83
84 - name: Remove Jenkins repo from main sources.list
85   lineinfile:
86     path: /etc/apt/sources.list
87     regexp: "deb .*pkg.jenkins.io.*"
88     state: absent
89
90 - name: Remove legacy Jenkins key and config
91   shell: |
92     gpg --batch --yes --delete-key "Jenkins Project" || true
93     rm -f /etc/apt/sources.list.d/jenkins*.list
94     rm -f /usr/share/keyrings/jenkins-keyring.asc
95     rm -f /etc/apt/trusted.gpg.d/jenkins.gpg
96     rm -f /etc/apt/trusted.gpg
97   ignore_errors: true
98
99 # Core tools
100 - name: Install core tools
101   apt:
102     name:
103       - curl
104       - gnupg
105       - ca-certificates
106       - apt-transport-https
107       - lsb-release
108     state: present
109
110 # Docker install
111 - name: Add Docker GPG key
112   apt_key:
113     url: https://download.docker.com/linux/ubuntu/gpg
114     state: present
115
116 - name: Add Docker apt repository
117   apt_repository:
118     repo: deb [arch=amd64] https://download.docker.com/linux/ubuntu {{ ansible_distribution_release }} stable
119     filename: docker
120     state: present
121
122 - name: Update apt cache
123   apt:
124     update_cache: yes
125
126 - name: Install Docker & dependencies
127   apt:
128     name:
129       - docker-ce
130       - docker-ce-cli
131       - containerd.io
132     state: present
133
134
135
136 - name: Enable and start Docker
137   systemd:
138     name: docker
139     enabled: true
140     state: started
141
142 - name: Add ubuntu user to docker group
143   user:
144     name: ubuntu
145     groups: docker
146     append: yes
147
148 # cri-dockerd setup
149 - name: Install cri-dockerd dependencies
150   apt:
151     name:
152       - git
153       - golang
154       - make
155     state: present
156
157 - name: Clone cri-dockerd (stable v0.3.1)
158   git:
159     repo: https://github.com/Mirantis/cri-dockerd.git
160     dest: /opt/cri-dockerd
161     version: v0.3.1
162
163 - name: Download and install Go 1.20.13
164   shell: |
165     wget https://go.dev/dl/go1.20.13.linux-amd64.tar.gz -O /tmp/go.tar.gz
166     rm -rf /usr/local/go && tar -C /usr/local -xzf /tmp/go.tar.gz
167     echo "export PATH=/usr/local/go/bin:$PATH" >> /etc/profile
168   args:
169     creates: /usr/local/go/bin/go
170
171 - name: Build cri-dockerd with Go 1.23+
172   shell: |
173     export PATH=$PATH:/usr/local/go/bin
174     cd /opt/cri-dockerd
175     mkdir -p bin
176     go build -o bin/cri-dockerd
177
178 - name: Move cri-dockerd binary
179   copy:
180     src: /opt/cri-dockerd/bin/cri-dockerd
181     dest: /usr/local/bin/cri-dockerd
182     mode: '0755'
183     remote_src: true
184
185 - name: Configure cri-dockerd systemd service
186   shell: |
187     cd /opt/cri-dockerd
188     cp packaging/systemd/* /etc/systemd/system/
189     sed -i 's:/usr/bin/cri-dockerd:/usr/local/bin/cri-dockerd:' /etc/systemd/system/cri-dockerd.service
190
191 - name: Enable and start cri-dockerd service
192   systemd:
193     name: cri-dockerd.service
194     daemon_reload: yes
195     enabled: yes
196     state: started
197
198 # Jenkins install
199 - name: Install Java 17 (Jenkins requirement)
200   apt:
201     name: openjdk-17-jdk
202     state: present
```

```

282 - name: Add Jenkins GPG key
283   get_url:
284     url: https://pkg.jenkins.io/debian/jenkins.io.key
285     dest: /usr/share/keyrings/jenkins-keyring.asc
286     mode: '0644'
287
288 - name: Add Jenkins apt repository
289   copy:
290     dest: /etc/apt/sources.list.d/jenkins.list
291     content: "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian binary/"
292     mode: '0644'
293
294 - name: Install Jenkins
295   apt:
296     name: jenkins
297     update_cache: yes
298     state: present
299
300 - name: Enable and start Jenkins
301   systemd:
302     name: jenkins
303     enabled: true
304     state: started
305
306 # Disable swap (K8s requirement)
307 - name: Disable swap
308   command: swapoff -a
309
310 - name: Comment swap in fstab
311   replace:
312     path: /etc/fstab
313     regexp: '^(\s*)\s\sswap(\s.*)\s$'
314     replace: '# \1'
315
316 # Kubernetes setup
317 - name: Create keyrings directory
318   file:
319     path: /etc/apt/keyrings
320     state: directory
321     mode: '0755'
322
323 - name: Download Kubernetes GPG key
324   shell: |
325     curl -fsSL https://pkgs.k8s.io/core/stable/v1.30/deb/Release.key | gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
326   args:
327     creates: /etc/apt/keyrings/kubernetes-apt-keyring.gpg
328
329 - name: Add Kubernetes apt repo
330   copy:
331     dest: /etc/apt/sources.list.d/kubernetes.list
332     content: "deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core/stable/v1.30/deb/ "
333     mode: '0644'
334
335 - name: Update apt cache for Kubernetes
336   apt:
337     update_cache: yes
338
339 - name: Install Kubernetes components
340   apt:
341     name:
342       - kubelet
343       - kubeadm
344       - kubectl
345     state: present
346
347 - name: Hold Kubernetes components
348   command: apt-mark hold kubelet kubeadm kubectl
349
350 - name: Initialize Kubernetes (using Docker)
351   command: kubeadm init --pod-network-cidr=192.168.0.0/16 --cri-socket=unix:///var/run/cri-dockerd.sock
352   args:
353     creates: /etc/kubernetes/admin.conf
354
355 # Kubeconfig setup
356 - name: Setup kubeconfig for ubuntu user
357   block:
358     - name: Create .kube directory for ubuntu
359       file:
360         path: /home/ubuntu/.kube
361         state: directory
362         owner: ubuntu
363         group: ubuntu
364         mode: '0755'
365
366     - name: Copy Kubernetes admin.conf to ubuntu kube config
367       copy:
368         src: /etc/kubernetes/admin.conf
369         dest: /home/ubuntu/.kube/config
370         owner: ubuntu
371         group: ubuntu
372         mode: '0644'
373         remote_src: true
374
375 - name: Install Calico CNI plugin
376   become: true
377   become_user: ubuntu
378   shell: kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.25.0/manifests/calico.yaml
379   environment:
380     KUBECONFIG: /home/ubuntu/.kube/config
381
382 - name: Ensure OPA namespace is created before installing Gatekeeper
383   become: true
384   become_user: ubuntu
385   shell: kubectl create namespace opa || true
386   environment:
387     KUBECONFIG: /home/ubuntu/.kube/config
388
389 - name: Install Prometheus with custom scrape config
390   become_user: ubuntu
391   shell: |
392     helm install prometheus prometheus-community/prometheus \
393       --namespace monitoring \
394       --create-namespace \
395       --set server.persistentVolume.enabled=false \
396       --set alertmanager.enabled=false \
397       --set server.podLabels.app=prometheus \
398       --set server.podLabels.com=prometheus \
399       --set-file server.extraScrapeConfigs=prometheus-additional-scrape-config.yaml
400
401 - name: Deploy OPA Gatekeeper
402   become: true
403   become_user: ubuntu
404   shell: kubectl apply --validate=false -f https://raw.githubusercontent.com/open-policy-agent/gatekeeper/master/deploy/gatekeeper.yaml
405   environment:
406     KUBECONFIG: /home/ubuntu/.kube/config
407

```

To run the Ansible script

- `ansible-playbook -i inventory/hosts.ini setup.yml`

3.3 Kubernetes

Kubernetes and its components including kubeadm and kubectl is configured using Ansible on the AWS EC2 instance to orchestrate containerized applications with a self-healing CI/CD pipeline. After running and configuring the EC2 instance, log in to it with your local system using Bash terminal or one of the tools like PuTTY. Upon successful installation, the Kubernetes cluster could be accessed using kubectl command at the terminal to manage our deployments, service and other Kubernetes cluster resources.

```
ubuntu@ip-10-0-1-62:~$ kubectl get pods
NAME                                READY   STATUS              RESTARTS   AGE
crashpod                             0/1     CrashLoopBackOff   88 (47s ago) 23h
failpod                              0/1     CrashLoopBackOff   86 (3m8s ago) 23h
failure-classification-29244765-nlwd6 0/1     Completed          0           6m16s
failure-classification-29244770-wjr8n 0/1     Completed          0           76s
failure-test-169z5                   0/1     Completed          0           8m21s
flask-app-67c44584cb-8rskh           1/1     Running             0           58m
flask-app-67c44584cb-ff955           1/1     Running             0           58m
flask-app-6b4d8f789b-2szsp           0/1     ImagePullBackOff   0           52m
health-scoring-29244765-kz4lh        0/1     Completed          0           6m16s
health-scoring-29244768-4gfps       0/1     Completed          0           3m16s
health-scoring-29244771-h7vgb       0/1     Completed          0           16s
postgres-59c4578695-hbm8z           0/1     Running             2 (3h4m ago) 23h
postgres-954f854b9-ntvfk            0/1     Running             2 (3h4m ago) 23h
trust-score-check-29244760-ql8zh     0/1     Completed          0           11m
trust-score-check-29244770-gl2cb     0/1     Completed          0           76s
trust-test-68h4v                     0/1     Completed          0           8m55s
ubuntu@ip-10-0-1-62:~$
```

- kubectl get pods
- kubectl apply -f <yaml>

3.4 Jenkins

Jenkins server is deployed on an AWS EC2 instance. After instance is launched and configured completely connect with your local system via Bash terminal or any other tool like PuTTY. Installation of Jenkins is done by Ansible and initiate the service. The web console of Jenkins can then be visited in through the browser using the Public IPv4 of the EC2 instance and appended with port 8080

```
ubuntu@ip-10-0-1-62:~$
* Documentation: https://help.ubuntu.com
* Management:   https://landscape.canonical.com
* Support:      https://ubuntu.com/advantage

System information as of Sat Aug 9 22:25:42 UTC 2025

System load: 0.39892578125   Users logged in: 0
Usage of /: 16.8% of 48.27GB   IPv4 address for docker0: 172.17.0.1
Memory usage: 59%           IPv4 address for ens5: 10.0.1.62
Swap usage: 0%              IPv4 address for tunl0: 192.168.49.66
Processes: 229

* Ubuntu Pro delivers the most comprehensive open source security and
  compliance features.

https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is not enabled.

96 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

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

Last login: Sat Aug 9 15:10:37 2025 from 109.76.97.237
ubuntu@ip-10-0-1-62:~$ jenkins --version
2.521
ubuntu@ip-10-0-1-62:~$
```

Fig1: Jenkins on EC2

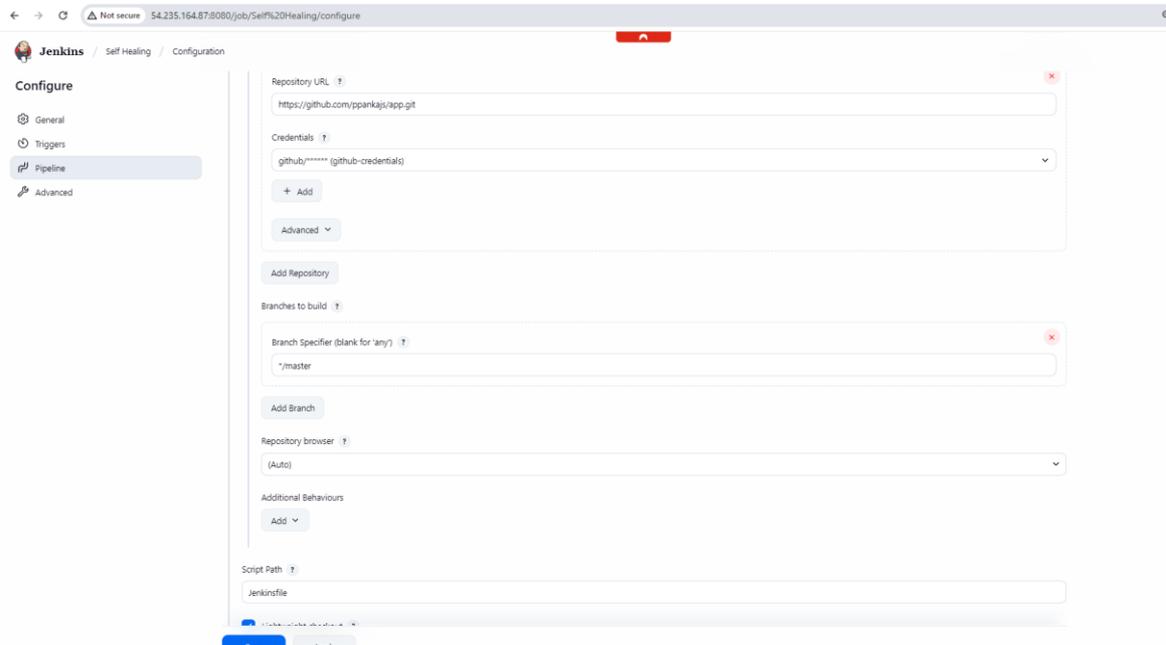


Fig2: Jenkins Pipeline Configuration

Jenkins File

```

1 pipeline {
2   agent any
3
4   environment {
5     IMAGE_NAME = "self-healing-app"
6     DOCKER_REGISTRY = "ppankajs"
7     TAG = "v${BUILD_NUMBER}"
8   }
9
10  stages {
11
12    stage('Clone Repo') {
13      steps {
14        git credentialsId: 'github-credentials', url: 'https://github.com/ppankajs/app.git'
15      }
16    }
17
18    stage('Build Docker Image') {
19      steps {
20        script {
21          docker.build("${DOCKER_REGISTRY}/${IMAGE_NAME}:${TAG}")
22        }
23      }
24    }
25
26    stage('Push Docker Image') {
27      steps {
28        script {
29          docker.withRegistry('', 'docker-credentials') {
30            docker.image("${DOCKER_REGISTRY}/${IMAGE_NAME}:${TAG}").push()
31          }
32        }
33      }
34    }
35
36    stage('Deploy PostgreSQL') {
37      steps {
38        sh 'kubectl apply -f k8s/postgres.yaml'
39      }
40    }
41  }

```

```

Jenkinsfile X
Jenkinsfile
1 pipeline {
10 stages {
42 stage('Deploy Flask App') {
43 steps {
44 // Replace image tag in deployment file before applying
45 sh """
46 sed 's|ppankajs/self-healing-app:latest|${DOCKER_REGISTRY}/${IMAGE_NAME}:${TAG}|' k8s/deployment.yaml | kubectl apply -f -
47 """
48 }
49 }
50
51 stage('Expose Flask Service') {
52 steps {
53 sh 'kubectl apply -f k8s/service.yaml'
54 // sh 'kubectl apply -f k8s/prometheus-additional-scrape-config.yaml'
55 }
56 }
57
58 stage('RBAC for Trust Check CronJob') {
59 steps {
60 sh 'kubectl apply -f k8s/trust-score-rbac.yaml'
61 sh 'kubectl apply -f k8s/failure-sa.yaml'
62 sh 'kubectl apply -f k8s/rollback-access.yaml'
63 sh 'kubectl apply -f k8s/rollback-access.yaml'
64 }
65 }
66
67 stage('Deploy CronJobs (Self-Healing)') {
68 steps {
69 script {
70 def cronJobs = [
71 "cron-failure-classification.yaml",
72 "cron-trust-score.yaml",
73 "cron-health-scoring.yaml",
74 "cron-rollback.yaml"
75 ]
76
77 for (cron in cronJobs) {
78 sh """
79 sed 's|ppankajs/self-healing-app:latest|${DOCKER_REGISTRY}/${IMAGE_NAME}:${TAG}|' k8s/cronjobs/${cron} | kubectl apply -f -
80 """
81 }
82 }
83 }
}

```

```

Jenkinsfile X
Jenkinsfile
1 pipeline {
10 stages {
84 }
85
86 stage('OPA: Apply Constraint Templates') {
87 steps {
88 sh '''
89 # Delete template if exists
90 kubectl delete constrainttemplate k8srequiredprobes --ignore-not-found
91
92 # Apply templates
93 kubectl apply -f opa/templates/label-template.yaml
94 kubectl apply -f opa/templates/probes-template.yaml
95
96 echo "Waiting for CRD 'k8srequiredprobes.constraints.gatekeeper.sh' to be registered and usable..."
97
98 # Wait loop for CRD to become available
99 for i in {1..12}; do
100 echo "Attempt $i: Checking if CRD exists and is ready..."
101 kubectl get crd k8srequiredprobes.constraints.gatekeeper.sh >/dev/null 2>&1 && break
102 sleep 5
103 done
104
105 # Final confirmation (with fail-safe)
106 if ! kubectl get crd k8srequiredprobes.constraints.gatekeeper.sh >/dev/null 2>&1; then
107 echo "CRD k8srequiredprobes.constraints.gatekeeper.sh' not found after waiting. Aborting."
108 exit 1
109 fi
110
111 echo "CRD is ready!"
112 '''
113 }
114 }
115
116 stage('OPA: Apply Constraints') {
117 steps {
118 sh '''
119 kubectl apply -f opa/label-constraint.yaml
120 kubectl apply -f opa/probes-constraint.yaml
121 '''
122 }
123 }
124
}

```

This is the JenkinsFile where my pipeline is written from fetching the code from github, building docker image, deploying my application as well as my cron jobs on the kubernetes.

3.5 Python Application

The Python Flask app is dockerized, and linked to a Postgres database. Flask app has health check, add users and list user endpoints. Kubernetes manifests will be generated over the application and database including Deployments, Services, ConfigMaps and Secrets. PostgreSQL accesses its data storage via PersistentVolumeClaim, both components are in the same cluster. After being deployed they can be accessed through the Kubernetes service, and all database queries are now handled by the internal PostgreSQL service.

```
app.py x
app.py > health
1 from flask import Flask, request, jsonify
2 import psycopg2
3 import socket
4 from retry import retry
5 from prometheus_flask_exporter import PrometheusMetrics
6
7 app = Flask(__name__)
8 metrics = PrometheusMetrics(app)
9
10 @retry(psycopg2.OperationalError, tries=3, delay=2)
11 def get_db_connection():
12     conn = psycopg2.connect(
13         host='db', # Use 'localhost' if you're not using Docker
14         # host='localhost',
15         database='self_healing',
16         user='postgres',
17         password='postgres'
18     )
19     return conn
20
21 @app.route('/')
22 def home():
23     return f"Hello from {socket.gethostname()}!"
24
25 @app.route('/health')
26 def health():
27     return "OK", 200
28
29 @app.route('/db-health')
30 def db_health():
31     try:
32         conn = get_db_connection()
33         conn.close()
34         return "DB UP", 200
35     except Exception as e:
36         return jsonify({"status": "DB DOWN", "error": str(e)}), 500
37
38 @app.route('/add', methods=['GET', 'POST'])
39
app.py x
app.py > health
1 from flask import Flask, request, jsonify
2 import psycopg2
3 import socket
4 from retry import retry
5 from prometheus_flask_exporter import PrometheusMetrics
6
7 app = Flask(__name__)
8 metrics = PrometheusMetrics(app)
9
10 @retry(psycopg2.OperationalError, tries=3, delay=2)
11 def get_db_connection():
12     conn = psycopg2.connect(
13         host='db', # Use 'localhost' if you're not using Docker
14         # host='localhost',
15         database='self_healing',
16         user='postgres',
17         password='postgres'
18     )
19     return conn
20
21 @app.route('/')
22 def home():
23     return f"Hello from {socket.gethostname()}!"
24
25 @app.route('/health')
26 def health():
27     return "OK", 200
28
29 @app.route('/db-health')
30 def db_health():
31     try:
32         conn = get_db_connection()
33         conn.close()
34         return "DB UP", 200
35     except Exception as e:
36         return jsonify({"status": "DB DOWN", "error": str(e)}), 500
37
38 @app.route('/add', methods=['GET', 'POST'])
```

```

EXPLORER
SELF_HEALING_FLASK_PIPELINE...
k8s
  cronjobs
    ! cron-failure-classification.yaml
    ! cron-health-scoring.yaml
    ! cron-rollback.yaml
    ! cron-trust-score.yaml
  deployment.yaml
  failure-sa.yaml
  postgres.yaml
  prometheus-additional-scrape-config.yaml
  rollback-access.yaml
  service.yaml
  trust-score-rbac.yaml
opa
  templates
    ! label-template.yaml
    ! probes-template.yaml
    ! label-constraint.yaml
  > scripts
  > venv
  app.py
  Dockerfile
  init_db.sql
  Jenkinsfile
  requirements.txt

app.py
health
1 from flask import Flask, request, jsonify
2 import psycopg2
3 import socket
4 from retry import retry
5 from prometheus_flask_exporter import PrometheusMetrics
6
7 app = Flask(__name__)
8 metrics = PrometheusMetrics(app)
9
10 Tabnine | Edit | Test | Explain | Document
11 @retry(psycopg2.OperationalError, tries=3, delay=2)
12 def get_db_connection():
13     conn = psycopg2.connect(
14         host="db", # Use 'localhost' if you're not using Docker
15         # host="localhost",
16         database="self_healing",
17         user="postgres",
18         password="postgres"
19     )
20     return conn
21
22 Tabnine | Edit | Test | Explain | Document
23 @app.route('/')
24 def home():
25     return f"Hello from {socket.gethostname()}!"
26
27 Tabnine | Edit | Test | Explain | Document
28 @app.route('/health')
29 def health():
30     return "OK", 200
31
32 Tabnine | Edit | Test | Explain | Document
33 @app.route('/db-health')
34 def db_health():
35     try:
36         conn = get_db_connection()
37         conn.close()
38         return "DB UP", 200
39     except Exception as e:
40         return jsonify({"status": "DB DOWN", "error": str(e)}), 500
41
42 Tabnine | Edit | Test | Explain | Document
43 @app.route('/add', methods=['GET', 'POST'])
44 def add_user():

```

```

EXPLORER
SELF_HEALING_FLASK_PIPELINE_BUNDLE
k8s
  cronjobs
    ! cron-failure-classification.yaml
    ! cron-health-scoring.yaml
    ! cron-rollback.yaml
    ! cron-trust-score.yaml
  deployment.yaml
  failure-sa.yaml
  postgres.yaml
  prometheus-additional-scrape-config.yaml
  rollback-access.yaml
  service.yaml
  trust-score-rbac.yaml
opa
  templates
    ! label-template.yaml
    ! probes-template.yaml
    ! label-constraint.yaml
    ! probes-constraint.yaml
  > scripts
  explainable_rollback.py
  failure_classification.py
  health_scoring.py
  trust_score.py
  > venv
  app.py
  Dockerfile
  init_db.sql
  Jenkinsfile
  requirements.txt

```

Fig4: Application Structure

3.6 Docker

Docker was deployed on the AWS EC2 instance via Ansible and Kubernetes was customized to manage containerized applications into the self-healing CI/CD process.

```
ubuntu@ip-10-0-1-62:~$ docker --version
Docker version 28.3.3, build 980b856
ubuntu@ip-10-0-1-62:~$
```

3.7 Scripts

Five Python scripts are Failure Classification, Trust-Score, Dynamic Policy Enforcement, Health Scoring, and Explainable Rollbacks that run in Kubernetes to detect issues, enforce policies, and trigger automated recovery. Each script has its own cron job.

```
explainable_rollback.py X failure_classification.py health_scoring.py trust_score.py
scripts > explainable_rollback.py > trigger_rollback
1 import json
2 import os
3 import subprocess
4
5 SCORE_THRESHOLD = 70
6 DATA_DIR = "/data"
7 ROLLBACK_TAG = "v96"
8
9 Tabnine | Edit | Test | Explain | Document
10 def read_score(file_name):
11     try:
12         with open(os.path.join(DATA_DIR, file_name), "r") as f:
13             data = json.load(f)
14             return list(data.values())[0]
15     except Exception as e:
16         print(f"[ERROR] Failed to read {file_name}: {e}")
17         return 0
18
19 Tabnine | Edit | Test | Explain | Document
20 def get_latest_scores():
21     scores = {
22         "failure": read_score("failure_score.json"),
23         "trust": read_score("trust_score.json"),
24         "health": read_score("health_score.json"),
25     }
26     print(f"[SCORES] {scores}")
27     return scores
28
29 Tabnine | Edit | Test | Explain | Document
30 def trigger_rollback():
31     print(f"[ROLLBACK] Deployment score too low. Rolling back to {ROLLBACK_TAG}")
32     subprocess.run(
33         ["kubectl", "set", "image", "deployment/flask-app",
34          f"flask-container-ppankajs/self-healing-app:{ROLLBACK_TAG}"]
35     )
36
37 Tabnine | Edit | Test | Explain | Document
38 def main():
39     scores = get_latest_scores()
40     avg_score = sum(scores.values()) // len(scores)
41     print(f"[EVALUATION] Deployment Score: {avg_score}")
42
43 explainable_rollback.py X failure_classification.py X health_scoring.py trust_score.py
scripts > failure_classification.py > classify_crashloopbackoff
1 import subprocess
2 import json
3 import os
4
5 output_file = "/data/failure_score.json"
6
7 Tabnine | Edit | Test | Explain | Document
8 def classify_crashloopbackoff():
9     print("[INFO] Starting Failure Classification...")
10    result = subprocess.run(
11        ["kubectl", "get", "pods", "-A", "-o", "json"],
12        capture_output=True, text=True
13    )
14
15    found_crash = False
16    try:
17        pods = json.loads(result.stdout)
18        for pod in pods["items"]:
19            statuses = pod.get("status", {}).get("containerStatuses", [])
20            for container in statuses:
21                waiting = container.get("state", {}).get("waiting", {})
22                # If waiting.get("reason") == "CrashLoopBackOff":
23                if waiting.get("reason") in ["CrashLoopBackOff", "ImagePullBackOff"]:
24                    print(f"[FAILURE] CrashLoopBackOff in {pod['metadata']['name']}")
25                    found_crash = True
26    except json.JSONDecodeError:
27        print("[ERROR] Failed to parse kubectl output.")
28
29    score = 30 if found_crash else 100
30    print(f"[FAILURE SCORE] {score}")
31    with open(output_file, "w") as f:
32        json.dump({"failure": score}, f)
33
34    if __name__ == "__main__":
35        classify_crashloopbackoff()
```

```

scripts > trust_score.py > evaluate_trust
1  import subprocess
2  import json
3  import sys
4  import os
5
6  TRUSTED_TAGS = ["v78", "v77", "v76", "v89"]
7  output_file = "/data/trust_score.json"
8
9  Tabnine | Edit | Test | Explain | Document
10 def evaluate_trust():
11     print("[INFO] Checking image trust score..")
12     result = subprocess.run(
13         ["kubectl", "get", "deployment", "flask-app", "-o", "json"],
14         capture_output=True, text=True
15     )
16     try:
17         deployment = json.loads(result.stdout)
18         image = deployment['spec']['template']['spec']['containers'][0]['image']
19         tag = image.split(":")[-1]
20         trust_score = 100 if tag in TRUSTED_TAGS else 40
21         print(f"[IMAGE] {image}")
22         print(f"[TRUST SCORE] {trust_score}")
23     except Exception as e:
24         print(f"[ERROR] Failed to evaluate trust: {e}")
25         trust_score = 40
26
27     with open(output_file, "w") as f:
28         json.dump({"trust": trust_score}, f)
29
30 if __name__ == "__main__":
31     evaluate_trust()
32

```

```

explainable_rollback.py  failure_classification.py  health_scoring.py ×  trust_score.py
scripts > health_scoring.py > ...
1  import random
2  import json
3  import os
4
5  output_file = "/data/health_score.json"
6
7  Tabnine | Edit | Test | Explain | Document
8  def get_mock_metrics():
9     cpu = random.randint(20, 95)
10    memory = random.randint(30, 95)
11    return cpu, memory
12
13 Tabnine | Edit | Test | Explain | Document
14 def calculate_health_score(cpu, memory):
15     return 100 - ((cpu + memory) // 2)
16
17 if __name__ == "__main__":
18     print("[INFO] Evaluating resource health score..")
19     cpu, mem = get_mock_metrics()
20     score = calculate_health_score(cpu, mem)
21     print(f"[HEALTH SCORE] {score}/100")
22     with open(output_file, "w") as f:
23         json.dump({"health": score}, f)

```

3.8 Cron Jobs

All the Python self-healing scripts will be scheduled as individual Kubernetes jobs running on a certain time frequency in order to measure some metrics, apply policies, and take recovery actions as needed.

```
k8s > cronjobs > ! cron-failure-classification.yaml
1  apiVersion: batch/v1
2  kind: CronJob
3  metadata:
4    name: failure-classification
5    labels:
6      app: self-healing
7      owner: prasanna
8  spec:
9    schedule: "**/5 * * * *"
10   jobTemplate:
11     spec:
12       template:
13         metadata:
14           labels:
15             app: self-healing
16             owner: prasanna
17         spec:
18           serviceAccountName: failure-classification-sa
19           containers:
20             - name: failure-check
21               image: ppankajs/self-healing-app:latest
22               command: ["python", "/app/scripts/failure_classification.py"]
23               volumeMounts:
24                 - name: shared-data
25                   mountPath: /data
26           restartPolicy: OnFailure
27           volumes:
28             - name: shared-data
29               persistentVolumeClaim:
30                 claimName: score-storage
31
```

```
k8s > cronjobs > ! cron-rollback.yaml
1  apiVersion: batch/v1
2  kind: CronJob
3  metadata:
4    name: explainable-rollback
5    labels:
6      app: self-healing
7      owner: prasanna
8  spec:
9    schedule: "**/15 * * * *"
10   jobTemplate:
11     spec:
12       template:
13         metadata:
14           labels:
15             app: self-healing
16             owner: prasanna
17         spec:
18           serviceAccountName: rollback-sa
19           containers:
20             - name: rollback
21               image: ppankajs/self-healing-app:latest
22               command: ["python", "/app/scripts/explainable_rollback.py"]
23               volumeMounts:
24                 - name: shared-data
25                   mountPath: /data
26           restartPolicy: OnFailure
27           volumes:
28             - name: shared-data
29               persistentVolumeClaim:
30                 claimName: score-storage
31
```

```

! cron-failure-classification.yaml | ! cron-health-scoring.yaml X | ! cron-rollback.yaml | ! cron-trust-
k8s > cronjobs > ! cron-health-scoring.yaml
1  apiVersion: batch/v1
2  kind: CronJob
3  metadata:
4    name: health-scoring
5    labels:
6      app: self-healing
7      owner: prasanna
8  spec:
9    schedule: "*/3 * * * *"
10   jobTemplate:
11     spec:
12       template:
13         metadata:
14           labels:
15             app: self-healing
16             owner: prasanna
17         spec:
18           containers:
19             - name: health-score
20               image: ppankajs/self-healing-app:latest
21               command: ["python", "/app/scripts/health_scoring.py"]
22               volumeMounts:
23                 - name: shared-data
24                   mountPath: /data
25           restartPolicy: OnFailure
26           volumes:
27             - name: shared-data
28               persistentVolumeClaim:
29                 claimName: score-storage
30
! cron-failure-classification.yaml | ! cron-health-scoring.yaml | ! cron-rollback.yaml
k8s > cronjobs > ! cron-trust-score.yaml
1  apiVersion: batch/v1
2  kind: CronJob
3  metadata:
4    name: trust-score-check
5    labels:
6      app: self-healing
7      owner: prasanna
8  spec:
9    schedule: "*/10 * * * *"
10   jobTemplate:
11     spec:
12       template:
13         metadata:
14           labels:
15             app: self-healing
16             owner: prasanna
17         spec:
18           serviceAccountName: trust-check-sa
19           containers:
20             - name: trust-score
21               image: ppankajs/self-healing-app:latest
22               command: ["python", "/app/scripts/trust_score.py"]
23               volumeMounts:
24                 - name: shared-data
25                   mountPath: /data
26           restartPolicy: OnFailure
27           volumes:
28             - name: shared-data
29               persistentVolumeClaim:
30                 claimName: score-storage
31

```

- kubectl apply -f k8s/cronjobs/cron-failure-classification.yaml
- kubectl apply -f k8s/cronjobs/cron-trust-score.yaml
- kubectl apply -f k8s/cronjobs/cron-health-scoring.yaml
- kubectl apply -f k8s/cronjobs/cron-rollback.yaml

3.9 Open Policy Agent

Open Policy Agent (OPA) Gatekeeper is placed in Kubernetes cluster to have control over various policies that work, such as mandatory labels, health probes, and secure container settings, so that non-compliant workloads could be avoided to be deployed.

```
! label-template.yaml X
opa > templates > ! label-template.yaml
1  apiVersion: templates.gatekeeper.sh/v1beta1
2  kind: ConstraintTemplate
3  metadata:
4    name: k8srequiredlabels
5    labels:
6      app: self-healing
7      owner: prasanna
8  spec:
9    crd:
10   spec:
11     names:
12       kind: K8sRequiredLabels
13   targets:
14     - target: admission.k8s.gatekeeper.sh
15       rego: |
16         package k8srequiredlabels
17
18         violation[{"
19           "msg": msg,
20           "details": {"missing_labels": missing}
21         }] {
22           provided := {label | input.review.object.metadata.labels[label]}
23           required := {"app", "owner"}
24           missing := required - provided
25           count(missing) > 0
26           msg := sprintf("Missing required labels: %v", [missing])
27         }
28
! probes-template.yaml X
opa > templates > ! probes-template.yaml
1  apiVersion: templates.gatekeeper.sh/v1beta1
2  kind: ConstraintTemplate
3  metadata:
4    name: k8srequiredprobes
5    labels:
6      app: self-healing
7      owner: prasanna
8  spec:
9    crd:
10   spec:
11     names:
12       kind: K8sRequiredProbes
13   targets:
14     - target: admission.k8s.gatekeeper.sh
15       rego: |
16         package k8srequiredprobes
17
18         violation[{"msg": msg}] {
19           container := input.review.object.spec.template.spec.containers[_]
20           not container.livenessProbe
21           msg := "Liveness probe is required"
22         }
23
24         violation[{"msg": msg}] {
25           container := input.review.object.spec.template.spec.containers[_]
26           not container.readinessProbe
27           msg := "Readiness probe is required"
28         }
29
```

- `kubectl apply -f opa/label-constraint.yaml`
- `kubectl apply -f opa/probes-constraint.yaml`

4 Testing and Outputs

4.1 Changing the image tag of flask application which is unknown or does not exist

```
Jenkinsfile | deployment.yaml X
k8s > ! deployment.yaml
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: flask-app
5  labels:
6    app: self-healing
7    owner: prasanna
8  spec:
9    replicas: 2
10   selector:
11     matchLabels:
12       app: self-healing
13       owner: prasanna
14   template:
15     metadata:
16       labels:
17         app: self-healing
18         owner: prasanna
19         prometheus: enabled
20     annotations:
21       prometheus.io/scrape: "true"
22       prometheus.io/path: "/metrics"
23       prometheus.io/port: "5000"
24     spec:
25       containers:
26       - name: flask-container
27         image: ppankajs/self-healing-app:v99999
28         ports:
29         - containerPort: 5000
30         livenessProbe:
31           httpGet:
32             path: /health
33             port: 5000
34           initialDelaySeconds: 5
35           periodSeconds: 10
36         readinessProbe:
37           httpGet:
38             path: /health
39             port: 5000
40           initialDelaySeconds: 5
41           periodSeconds: 10
```

4.2 Getting all pods present in the Kubernetes cluster

```
ubuntu@ip-10-0-1-62:~$ kubectl get pods
NAME                                READY   STATUS              RESTARTS   AGE
crashpod                             0/1    CrashLoopBackOff   88 (47s ago)  23h
failpod                               0/1    CrashLoopBackOff   86 (3m8s ago)  23h
failure-classification-29244765-nlwd6 0/1    Completed           0             6m16s
failure-classification-29244770-wjr8n 0/1    Completed           0             76s
failure-test-l69z5                   0/1    Completed           0             8m21s
flask-app-67c44584cb-8rskh            1/1    Running             0             58m
flask-app-67c44584cb-ff955            1/1    Running             0             58m
flask-app-6b4d8f789b-2szsp            0/1    ImagePullBackOff    0             52m
health-scoring-29244765-kz4lh         0/1    Completed           0             6m16s
health-scoring-29244768-4gfps         0/1    Completed           0             3m16s
health-scoring-29244771-h7vgb         0/1    Completed           0             16s
postgres-59c4578695-hbm8z            0/1    Running             2 (3h4m ago)  23h
postgres-954f854b9-ntvfk              0/1    Running             2 (3h4m ago)  23h
trust-score-check-29244760-ql8zh      0/1    Completed           0             11m
trust-score-check-29244770-gl2cb      0/1    Completed           0             76s
trust-test-68h4v                       0/1    Completed           0             8m55s
ubuntu@ip-10-0-1-62:~$
```

After changing the Image tag of the flask application it is showing me the error called ImagePullBackoff Error. Crashpod and failpod are those used for testing my scripts sample pods. The command used for fetching all pods in Kubernetes

- kubectl get pods

4.3 Running the cron job

```
ubuntu@ip-10-0-1-62:~$ kubectl create job --from=cronjob/failure-classification failure-test
job.batch/failure-test created
ubuntu@ip-10-0-1-62:~$ kubectl create job --from=cronjob/trust-score-check trust-test
job.batch/trust-test created
ubuntu@ip-10-0-1-62:~$ kubectl create job --from=cronjob/health-scoring health-test
job.batch/health-test created
```

The command use for creating or running the cron job is

- `kubectl create job --from=cronjob/failure-classification failure-test`
- `kubectl create job --from=cronjob/trust-score-check trust-test`
- `kubectl create job --from=cronjob/health-scoring health-test`

The logs of created jobs are fetched from this command and here you can see the image which we have added in the flask app which is untrusted.

```
ubuntu@ip-10-0-1-62:~$ kubectl logs job/failure-test
[INFO] Starting Failure Classification...
[FAILURE] CrashLoopBackOff in crashpod
[FAILURE] CrashLoopBackOff in failpod
[FAILURE] CrashLoopBackOff in flask-app-6b4d8f789b-2szsp
[FAILURE SCORE] 30
ubuntu@ip-10-0-1-62:~$ kubectl logs job/trust-test
[INFO] Checking image trust score...
[IMAGE] ppankajs/self-healing-app:v99999
[TRUST SCORE] 40
ubuntu@ip-10-0-1-62:~$ kubectl logs job/health-test
[INFO] Evaluating resource health score...
[HEALTH SCORE] 48/100
ubuntu@ip-10-0-1-62:~$
```

- `kubectl logs job/failure-test`
- `kubectl logs job/trust-test`
- `kubectl logs job/health-test`

This the cron job for using the trusted image tag which is secure and the command use is

```
ubuntu@ip-10-0-1-62:~$ kubectl create job --from=cronjob/explainable-rollback rollback-test
job.batch/rollback-test created
```

- `kubectl create job --from=cronjob/explainable-rollback rollback-test`

The log of created job are fetched from this command

```
ubuntu@ip-10-0-1-62:~$ kubectl logs job/rollback-test
deployment.apps/flask-app image updated
[SCORES] {'failure': 30, 'trust': 40, 'health': 51}
[EVALUATION] Deployment Score: 40
[TRIGGERED] Score unhealthy. Triggering rollback...
[ROLLBACK] Deployment score too low. Rolling back to v96
```

- `kubectl logs job/rollback-test`

4.4 Output are monitor from the Prometheus

- The output after changing the image tag which is insecure and it shows that the application is down

The screenshot shows the Prometheus 'Status > Target health' page. It displays a list of targets under the 'kubernetes-service-endpoints' section. One target, `http://192.168.49.83:5000/metrics`, is marked as 'DOWN'. The error message below it reads: 'Error scraping target: Get "http://192.168.49.83:5000/metrics": dial tcp 192.168.49.83:5000: connect: connection refused'. Other targets are marked as 'UP'.

- After running all the scripts and applying the rollback, the application is up

The screenshot shows the Prometheus 'Status > Target health' page after a rollback. All targets are now marked as 'UP'. The 'kubernetes-service-endpoints' section shows 4/4 targets up, and the 'prometheus' section shows 1/1 target up. The error messages from the previous screenshot are no longer present.

5 References

HashiCorp, 2025. *Terraform Documentation*. [online] Available at: <https://developer.hashicorp.com/terraform/docs>

Ansible Project, 2025. *Ansible Documentation*. [online] Available at: <https://docs.ansible.com/>

Jenkins Project, 2025. *Jenkins User Documentation*. [online] Available at: <https://www.jenkins.io/doc/>

Docker Inc., 2025. *Docker Documentation*. [online] Available at: <https://docs.docker.com/>

Amazon Web Services, 2025. *Amazon EC2 Documentation*. [online] Available at: <https://docs.aws.amazon.com/ec2/>