

E-Wallet App Deployment Instructions and Results

A. Setting Up the Registry and Kubernetes Cluster

```
# container registry
You, 3 days ago | 1 author (You)
resource "digitalocean_container_registry" "ewallet-registry" {
  name           = var.registry_name
  subscription_tier_slug = var.registry_tier
  region         = var.region
}

# kubernetes cluster
You, 32 minutes ago | 1 author (You)
resource "digitalocean_kubernetes_cluster" "ewallet-cluster" {
  name       = var.cluster_name
  region    = var.region
  version   = var.k8s_version

  registry_integration = true

  You, 32 minutes ago | 1 author (You)
  node_pool {
    name       = "worker-node-pool"
    size       = var.node_size
    node_count = var.node_count
    auto_scale = false
  }
  You, 3 days ago • Update: terraform added
}

# ambil docker credentials untuk login registry
You, 3 days ago | 1 author (You)
resource "digitalocean_container_registry_docker_credentials" "creds" {
  registry_name = digitalocean_container_registry.ewallet-registry.name
  expiry_seconds = 86400
  write         = true
}
```

1. Create the registry and cluster with Terraform and split the code with several files in order to make it easy to control.
2. Create providers.tf to declare the platform is DigitalOcean. Also, Terraform Cloud is used for the integration with the pipeline in GitHub and embedded in this file for the integration.
3. Next, create main.tf to configure the registry and cluster. All the hardcode which is used for the registry and cluster, saved it in another file and it called variables.tf for the initial variables. Also, write the code for calling the docker credential in order to integrate with the pipeline in GitHub.

4. Create the environments folder, write it in dev.tfvars and save it for the custom variable that is used in main.tf.
5. Create an outputs.tf file to get the credentials such as registry_endpoint, kubeconfig, and docker_credentials.
6. Add the command below to execute inside the pipeline:

```
terraform init
```

```
terraform apply -var-file=environments/dev.tfvars
```

```
terraform output -raw kubeconfig > kubeconfig  
export KUBECONFIG=$(pwd)/kubeconfig
```

B. Build the docker image

```
You, 20 hours ago | 1 author (You)
FROM golang:1.25-alpine AS builder (last pushed 2 weeks ago)

# Set working directory di dalam container
WORKDIR /build

# Copy file dependency
COPY src/go.mod src/go.sum ./
RUN go mod download && go mod verify

COPY src/ .

ENV CGO_ENABLED=0

# Build aplikasi
RUN go build -v -o app .

FROM alpine:3.20 (last pushed 2 months ago)

WORKDIR /app

ENV GIN_MODE=release

COPY --from=builder /build/app .    You, 3 days ago • Init: E-wallet App

EXPOSE 8080

CMD ["/app"]
```

1. Build with the multi-stage schema so it would be lighter.
2. Don't forget to include CGO_ENABLED = 0 to get a smaller size because it won't save the C Library inside the image.
3. Add environment mode to release with GIN_MODE.
4. Build and push the image to the registry within the pipeline. The command for build and push the image to registry is below:

```
docker buildx build --platform linux/amd64 -t
registry.digitalocean.com/ewallet-registry/ewallet-app:2.4 . --push
```

5. Before moving to Kubernetes Cluster, the cluster used DigitalOcean with linux/amd64 platform. Adjust the image with linux/amd64 platform in order to use the image inside the cluster.

C. Applying The YML Manifests

1. Install the metrics-server to catch the usage from the resource cpu or memory. Also, I did the tuning resource for ds-cilium and coredns pods to get the light resource.

```
kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml

kubectl apply -k deploy/k8s/resource-tuning

kubectl -n kube-system patch ds cilium \
  --type='json' \

-p='[{"op":"add","path":"/spec/template/spec/containers/0/resources","value":{"requests":{"cpu":"100m","memory":"128Mi"},"limits":{"cpu":"200m","memory":"256Mi"}}}]'

kubectl -n kube-system patch deployment coredns \
  --type='json' \

-p='[{"op":"add","path":"/spec/template/spec/containers/0/resources","value":{"requests":{"cpu":"50m","memory":"70Mi"},"limits":{"cpu":"100m","memory":"150Mi"}}}]'
```

2. Created the folder for base setup and database. The base folder is to save the secret file such as the environment and also create the namespace. Inside the database, I created statefulset and PVC to implement the database and save the file inside kubernetes storage. I also insert the dummy to cluster.

```
kubectl apply -f deploy/k8s/base
kubectl apply -f deploy/k8s/database

kubectl cp src/db/db-money_movement.sql
ewallet/postgres-0:/tmp/db-money_movement.sql -n ewallet

kubectl exec -it -n ewallet postgres-0 -- sh
```

Inside the bash, login to database postgres and move to money_movement database. After that, insert this query:

```
CREATE EXTENSION IF NOT EXISTS "uuid-oss";
```

And then, quit the postgresql and insert this command while inside the bash.

```
psql -U postgres -d money_movement -f /tmp/db-money_movement.sql
```

Finally, we can crosscheck as well inside the postgresql and run the "\dt" command inside the money_movement database. You will see the dummy table.

3. Apply the application ewallet pods.

```
kubectl apply -f deploy/k8s/app/ewallet.yml
```

4. Implement the Prometheus and Grafana with a helm chart. Also, created the custom helm chart to implement the alert in order to send the alert to API Slack. And then apply the alert-cpu and configmap to visualize the dashboard. (user = admin, password = admin)

```
helm repo add prometheus-community  
https://prometheus-community.github.io/helm-charts  
helm repo update  
  
helm install prometheus prometheus-community/kube-prometheus-stack \\\n  -n monitoring --create-namespace \\\n  -f deploy/helm/monitoring/values.yml  
  
kubectl apply -f deploy/k8s/monitoring
```

5. Implement the ingress-nginx with helm chart and custom values. Make sure to activate the service monitor in order to give the data to prometheus and will be visualized in Grafana.

```
helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
```

```
helm repo update
helm install ingress-nginx ingress-nginx/ingress-nginx \
  --namespace ingress-nginx \
  --create-namespace \
  -f deploy/helm/ingress/values.yml
```

6. Add the DNS with sync the A record to personal DNS. This is how to get the external IP address.

```
kubectl get svc -n ingress-nginx
```

7. Install the cert manager to make the http to https in order to encrypt the data while doing request and response.

```
helm repo add jetstack https://charts.jetstack.io
helm repo update

helm install cert-manager jetstack/cert-manager \
  --namespace cert-manager \
  --create-namespace \
  --version v1.14.4 \
  --set installCRDs=true

kubectl apply -f deploy/k8s/ingress/cluster-issuer.yml
```

8. Create the ingress.yml to make the dns that already configured is available with connecting the app service inside the ingress file. Also, the ingress here is load balancer type.

```
kubectl apply -f deploy/k8s/ingress/ingress.yml
```

This is how to make sure the service is connected with the DNS.

```
kubectl get ingress -n ewallet
```

9. Apply the HPA in order to scale if the CPU resource usage is more than 70% request resource. But, according to the technical detail, there is one more scaling

with KEDA to scale based on the event driven. The system will scale up if the request per second is more than 50 per pod in 1 minute. So, if we want to implement HPA for both of them, we must only use KEDA for one deployment in order to prevent the conflict.

```
helm repo add kedacore https://kedacore.github.io/charts
helm repo update

helm install keda kedacore/keda \
  --namespace keda --create-namespace \
  -f deploy/helm/keda/values.yml

kubectl apply -f deploy/k8s/autoscaling
```

10. Create the logging with ELK stack and filebeat to collect the log each node. Implement the code below.

```
helm repo add elastic https://helm.elastic.co
helm repo update

helm install elasticsearch elastic/elasticsearch \
  --namespace logging --create-namespace \
  -f deploy/helm/elasticsearch/values.yml

helm install logstash elastic/logstash \
  --namespace logging \
  -f deploy/helm/logstash/values.yml

helm install filebeat elastic/filebeat \
  --namespace logging \
  -f deploy/helm/filebeat/values.yml

helm install kibana elastic/kibana \
  --namespace logging \
  -f deploy/helm/kibana/values.yml
```

11. Run this command:

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

After that, you can login to Grafana -> username = admin, password = admin

12. Run this command:


```
kubect1 port-forward svc/kibana-kibana -n logging 8090:5601
```

After that, you can login to Kibana -> username = elastic, password = changepw
Don't forget to set up the Discover inside Kibana to get the data from the index that is already defined in filebeat and Logstash.

D. Screenshoots/Links

1. Kubernetes Dashboards

← Back to Clusters

 **ewallet-cluster**
in [first-project](#) / SGP1 - 1.33.1-do.4

Actions ▾

Overview Resources Insights Marketplace Settings

Overview Learn

CONFIGURATION

CPU	Memory	Disk
6 vCPUs	12 GB	240 GB

Download Config File ⬇

Remind me how to use this file to connect to the cluster

NODE POOL STATUS

3/3 Running

worker-node-pool 3/3 nodes running

[View resource details](#)

NETWORKING

VPC network


default-sgp1

Pod and service network peering is available for clusters created on version 1.31 or

CONTROL PLANE

We recommend high availability to increase peace of mind for production workloads.

High availability

 **ewallet-cluster**
in [first-project](#) / SGP1 - 1.33.1-do.4


Actions ▾

Overview Resources Insights Marketplace Settings




Resources Learn

NODE POOLS

Filters + Add Node Pool

Name ▾	Created	Tags	Nodes	Autoscale
 worker-node-pool s-2vcpu-4gb - 1.33.1-do.4	5 hours ago	k8s k8s:6c431f64... k8s:worker	3	Off

NODES

 worker-node-pool-p3thn	5 hours ago
 worker-node-pool-p3th3	5 hours ago
 worker-node-pool-p3th8	5 hours ago

1-3 of 3 < >

[← Back to Clusters](#)



ewallet-cluster

in [first-project](#) / SGP1 - 1.33.1-do.4

Actions ▾

[Overview](#)

[Resources](#)

[Insights](#)

[Marketplace](#)

[Settings](#)

Insights

[Learn](#)

Select Object

ewallet-cluster ▾

Select Period

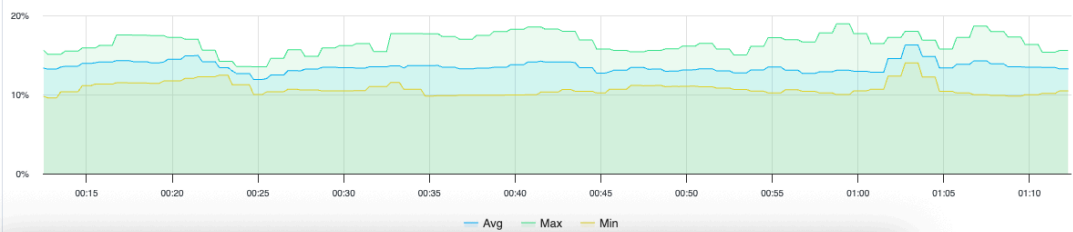
1 hour ▾

[Get Advanced Metrics](#) ▾

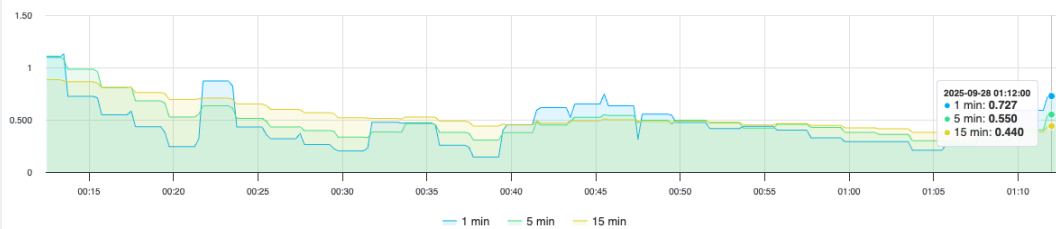
[Manage Resource Alerts](#)

Droplets

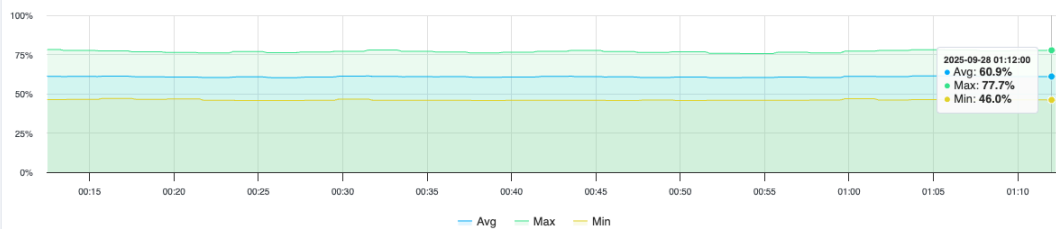
CPU Usage



Load Average



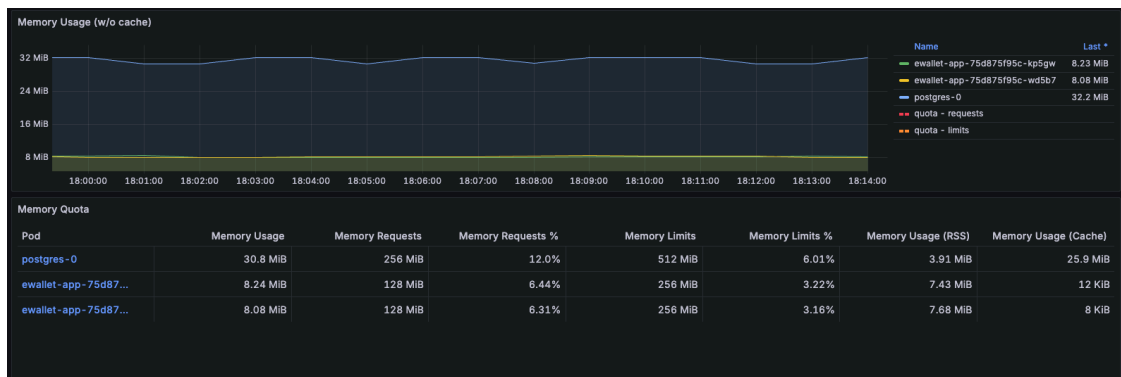
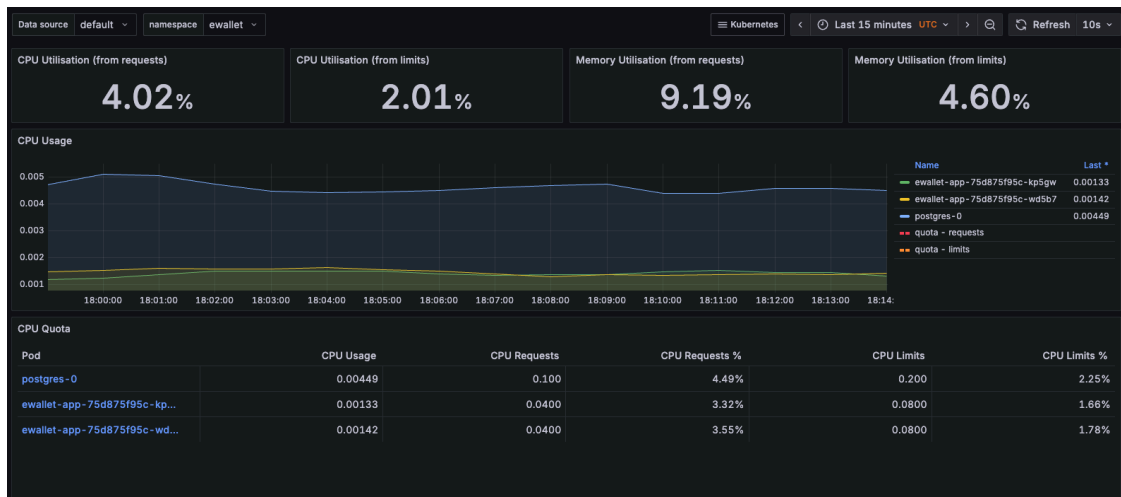
Memory Usage



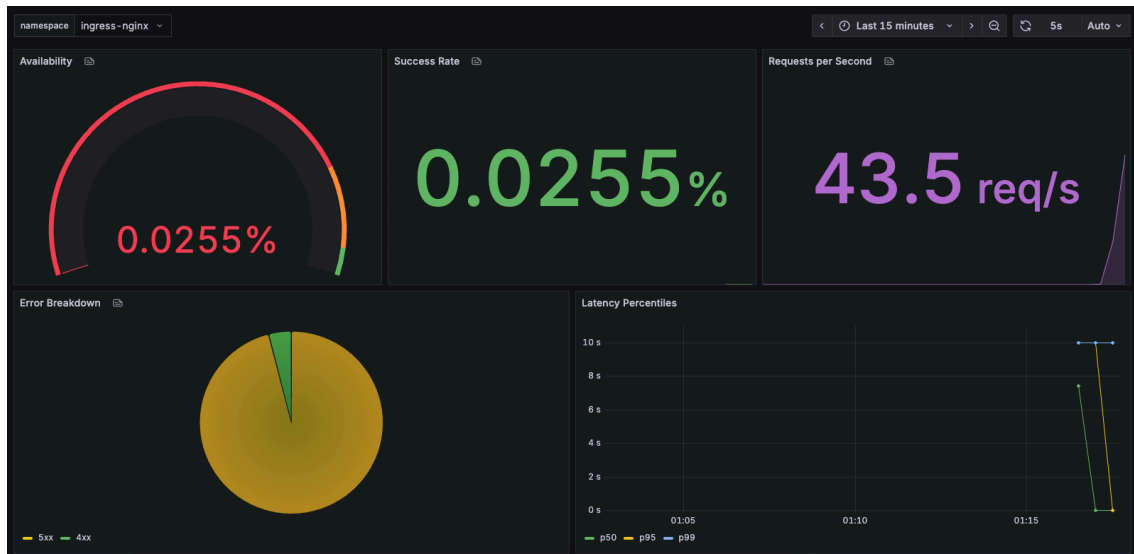


2. Monitoring (Grafana Dashboards)

Pod resource usage

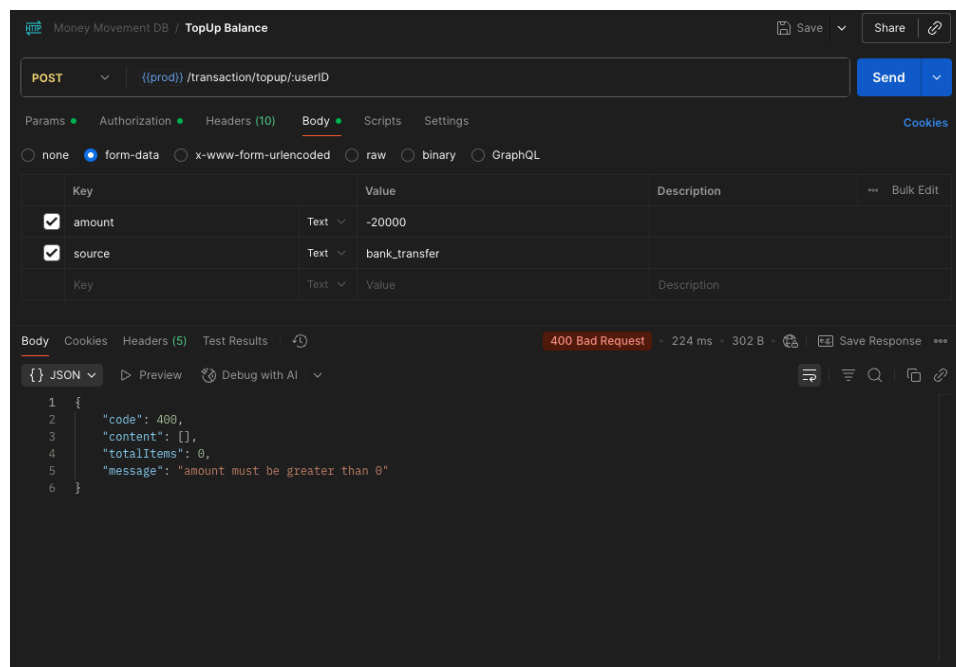


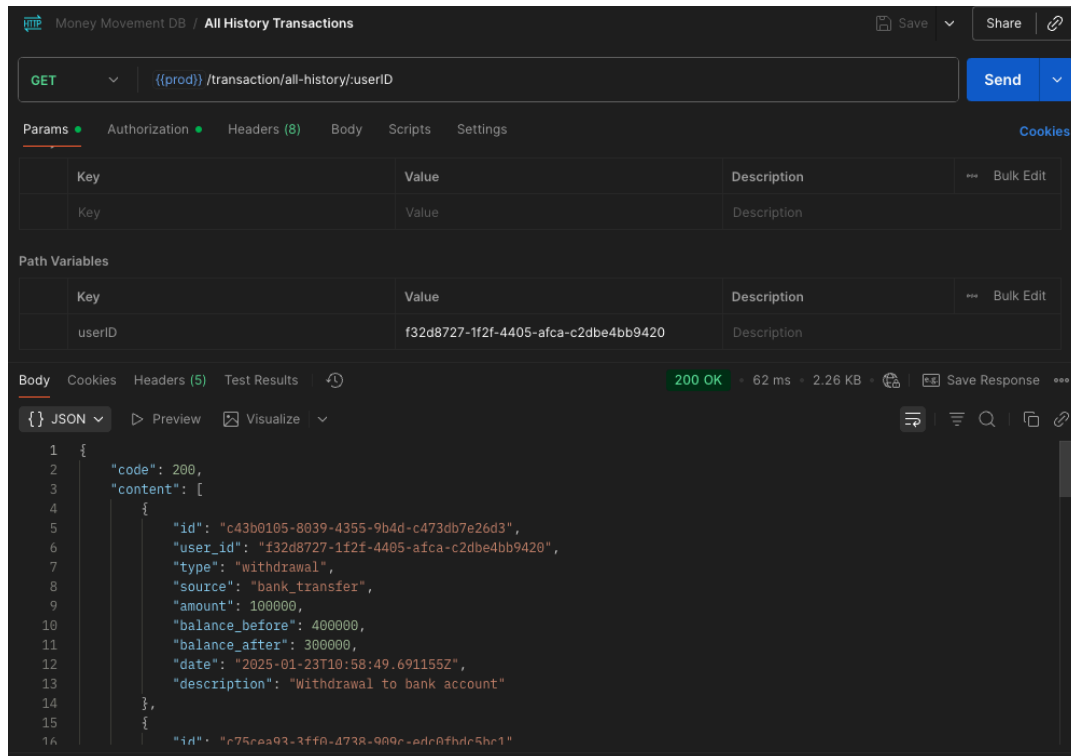
Application Health



3. Running Application (<https://books-api.site>)

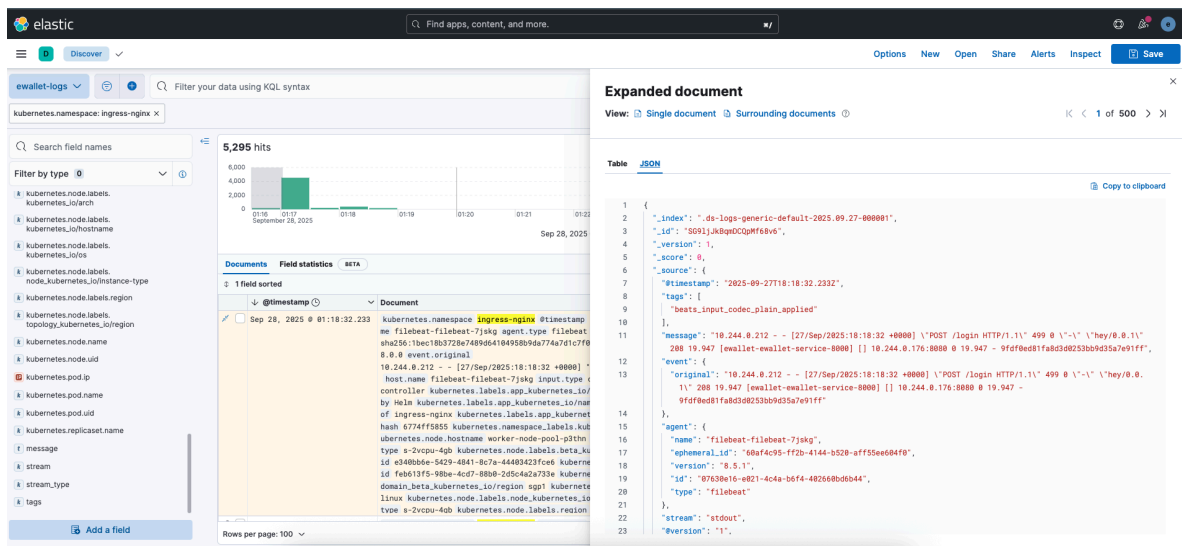
If you want to check API collection, it is located inside the repository “src/postman_collection/Money Movement DB.postman_collection.json”. Don't forget there is an authentication (JWT) for each of request except login.



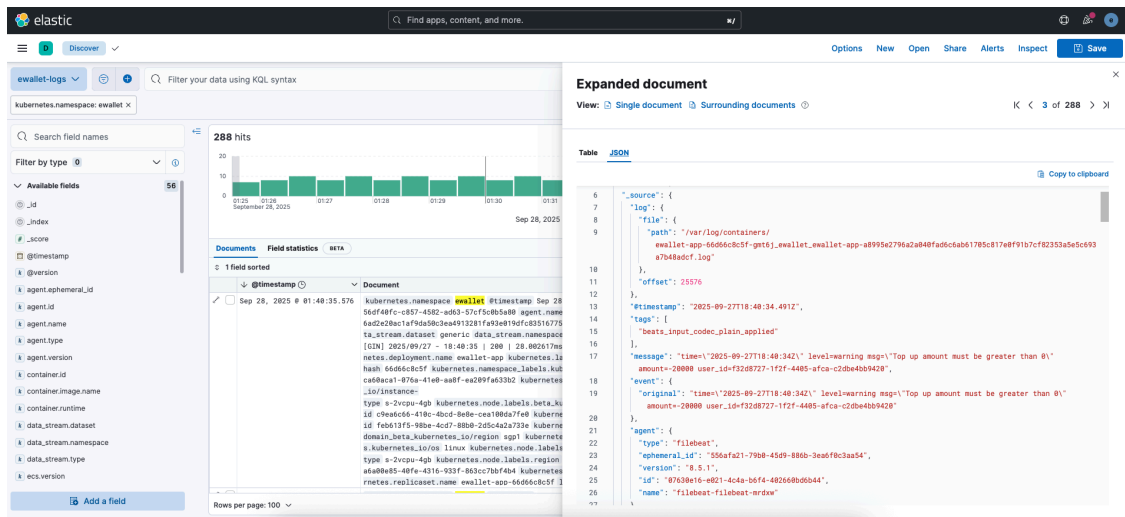


4. Logging

Requests Detail (in message JSON)



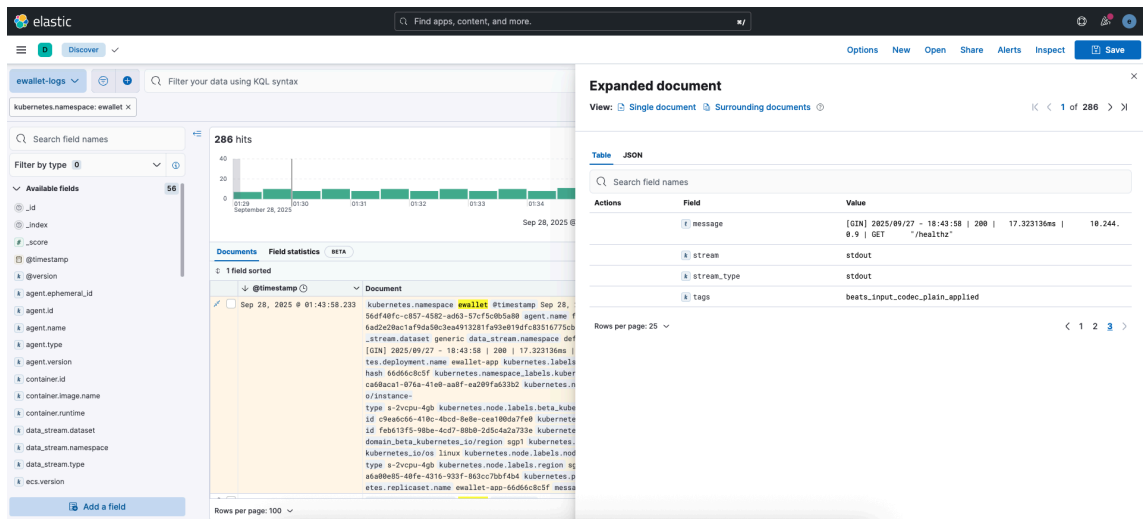
Errors (in message JSON)



The screenshot shows the Elastic UI interface. On the left, the search bar contains 'ewallet-logs' and the filter 'kubernetes.namespace: ewallet'. The search results show 288 hits. The expanded document view shows a JSON message with a 'message' field containing a log entry. The log entry is a warning message about a top amount being greater than 0.

```
6 {
7   "log": {
8     "file": {
9       "path": "/var/log/containers/
10       ewallet-app-6d66c8cf-get6j.ewallet.ewallet-app-a8995e2796a2a40fadc6ab61785c817e8f91b7f82353a5e5c93
11       a7b48dcf.log"
12     },
13     "offset": 25576
14   },
15   "timestamp": "2025-09-27T18:48:34.491Z",
16   "tags": [
17     "beats_input_codec_plain_applied"
18   ],
19   "message": "time=\"2025-09-27T18:48:34Z\" level=warning msg=\"Top up amount must be greater than 0\"
20   amount=20000 user_id=f32d8727-1f2f-4485-afca-c3db4e4b9428",
21   "event": {
22     "type": "filebeat",
23     "original": "time=\"2025-09-27T18:48:34Z\" level=warning msg=\"Top up amount must be greater than 0\"
24     amount=20000 user_id=f32d8727-1f2f-4485-afca-c3db4e4b9428"
25   },
26   "agent": {
27     "type": "filebeat",
28     "ephemeral_id": "555afa21-7908-45d9-8860-3e0dfbc3aa54",
29     "version": "8.5.1",
30     "id": "07630e16-e821-4c4a-b6f4-482668b00444",
31     "name": "filebeat-filebeat-nrdw"
32   }
33 }
```

Application Health (There is a /healthz API)



The screenshot shows the Elastic UI interface. On the left, the search bar contains 'ewallet-logs' and the filter 'kubernetes.namespace: ewallet'. The search results show 286 hits. The expanded document view shows a JSON message with a 'message' field containing a log entry. The log entry is a warning message about a top amount being greater than 0.

```
6 {
7   "log": {
8     "file": {
9       "path": "/var/log/containers/
10       ewallet-app-6d66c8cf-get6j.ewallet.ewallet-app-a8995e2796a2a40fadc6ab61785c817e8f91b7f82353a5e5c93
11       a7b48dcf.log"
12     },
13     "offset": 25576
14   },
15   "timestamp": "2025-09-27T18:48:34.491Z",
16   "tags": [
17     "beats_input_codec_plain_applied"
18   ],
19   "message": "time=\"2025-09-27T18:48:34Z\" level=warning msg=\"Top up amount must be greater than 0\"
20   amount=20000 user_id=f32d8727-1f2f-4485-afca-c3db4e4b9428",
21   "event": {
22     "type": "filebeat",
23     "original": "time=\"2025-09-27T18:48:34Z\" level=warning msg=\"Top up amount must be greater than 0\"
24     amount=20000 user_id=f32d8727-1f2f-4485-afca-c3db4e4b9428"
25   },
26   "agent": {
27     "type": "filebeat",
28     "ephemeral_id": "555afa21-7908-45d9-8860-3e0dfbc3aa54",
29     "version": "8.5.1",
30     "id": "07630e16-e821-4c4a-b6f4-482668b00444",
31     "name": "filebeat-filebeat-nrdw"
32   }
33 }
```

5. Alert (API Slack)

The screenshot shows a Slack interface for the workspace 'abdingara-org'. The channel is '#alertmanager-ewallet-app'. The left sidebar shows a list of channels, with '#alertmanager-ewallet-app' selected. The main content area displays a series of messages from the 'alertmanager-ewallet' app. The first message is a green checkmark indicating a resolved alert: 'Description: alertmanager monitoring/alertmanager-prometheus-kube-prometheus-alertmanager-0 failed to send 20% of notifications to slack. Resolved at: 2025-09-27 21:32:25.069 +0000 UTC'. The second message is a red warning icon followed by '[firing] AlertmanagerFailedToSendAlerts - warning'. It details an alert: 'Alert: AlertmanagerFailedToSendAlerts', 'Namespace: monitoring', 'Pod: alertmanager-prometheus-kube-prometheus-alertmanager-0', 'Status: FIRING', and 'Severity: warning'. The description states: 'Alertmanager monitoring/alertmanager-prometheus-kube-prometheus-alertmanager-0 failed to send 75% of notifications to slack. Started at: 2025-09-27 21:39:25.069 +0000 UTC'. The third message is a green checkmark indicating a resolved alert: 'Description: Alertmanager monitoring/alertmanager-prometheus-kube-prometheus-alertmanager-0 failed to send 50% of notifications to slack. Resolved at: 2025-09-27 21:58:55.069 +0000 UTC'. The fourth message is a red warning icon followed by '[firing] KubeMemoryOvercommit - warning'. It details an alert: 'Alert: KubeMemoryOvercommit', 'Namespace: N/A', 'Pod: N/A', 'Status: FIRING', and 'Severity: warning'. The description states: 'Cluster has overcommitted memory resource requests for Pods by 2.497G bytes and cannot tolerate node failure. Started at: 2025-09-27 22:06:05.553 +0000 UTC'. The bottom of the screen shows a status bar with 'AI is turned on for abdingara-org. Learn more'.