goto; amsterdam

Please

Ask questions through the app

Rate Session

Thank you!

Microservice Message Routing on Kubernetes

Frans van Buul

AxonIQ

Introductions

About me

- Evangelist at AxonIQ, the new company around Axon Framework.
- Prior roles
  - Presales architect for Hewlett Packard / Fortify
  - Java developer at Trifork and others
  - Security auditor at PwC

About this presentation

- Why microservices? Why Kubernetes?
- A tiny sample app.
- Splitting that up and deploying it as message-exchanging microservices on Kubernetes.

Why microservices? (and why not a simple monolith?)

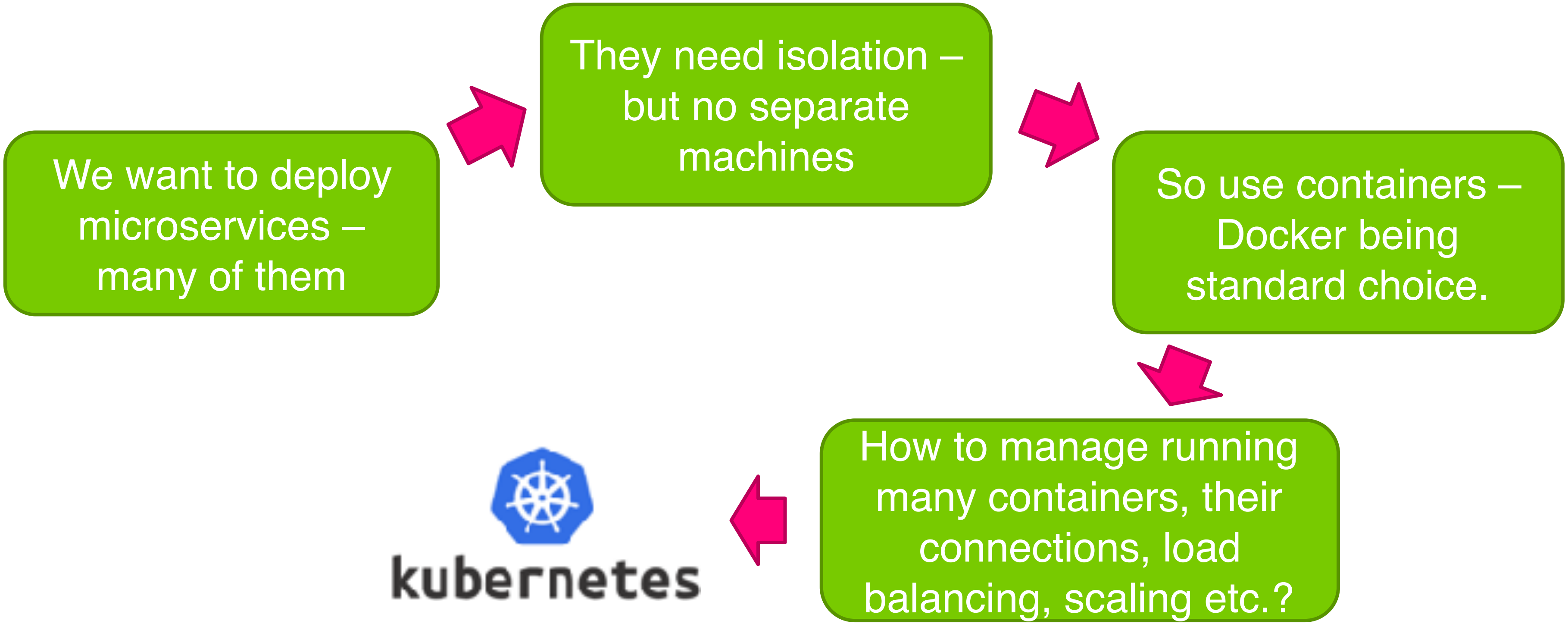
Agility

- Deploy new business functionality rapidly.
- Agile, Scrum, DevOps, ….

Scalability

- Have the ability to scale out as business demand grows.

Why Kubernetes?



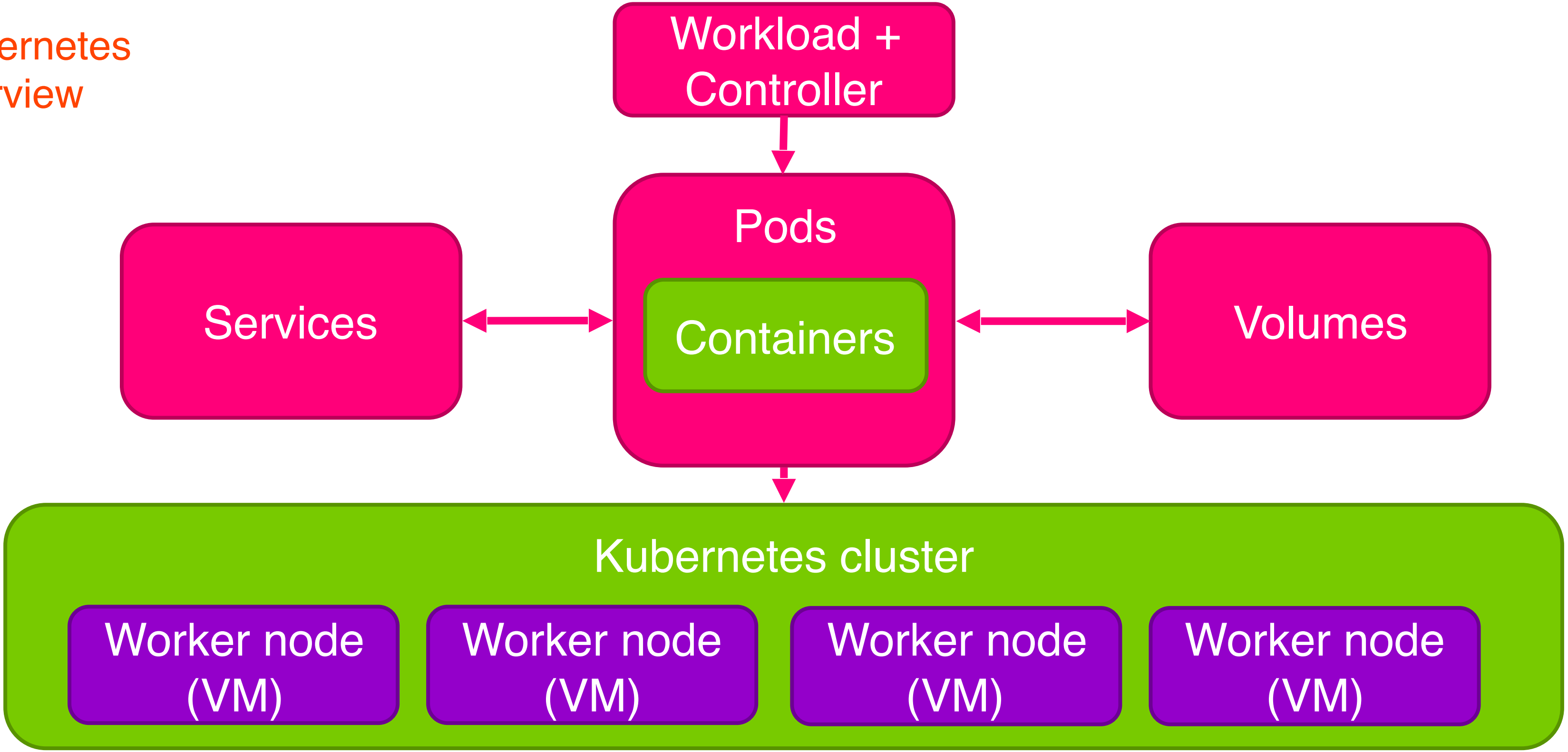
We want to deploy microservices – many of them

They need isolation – but no separate machines

So use containers – Docker being standard choice.

How to manage running many containers, their connections, load balancing, scaling etc.?

Kubernetes overview

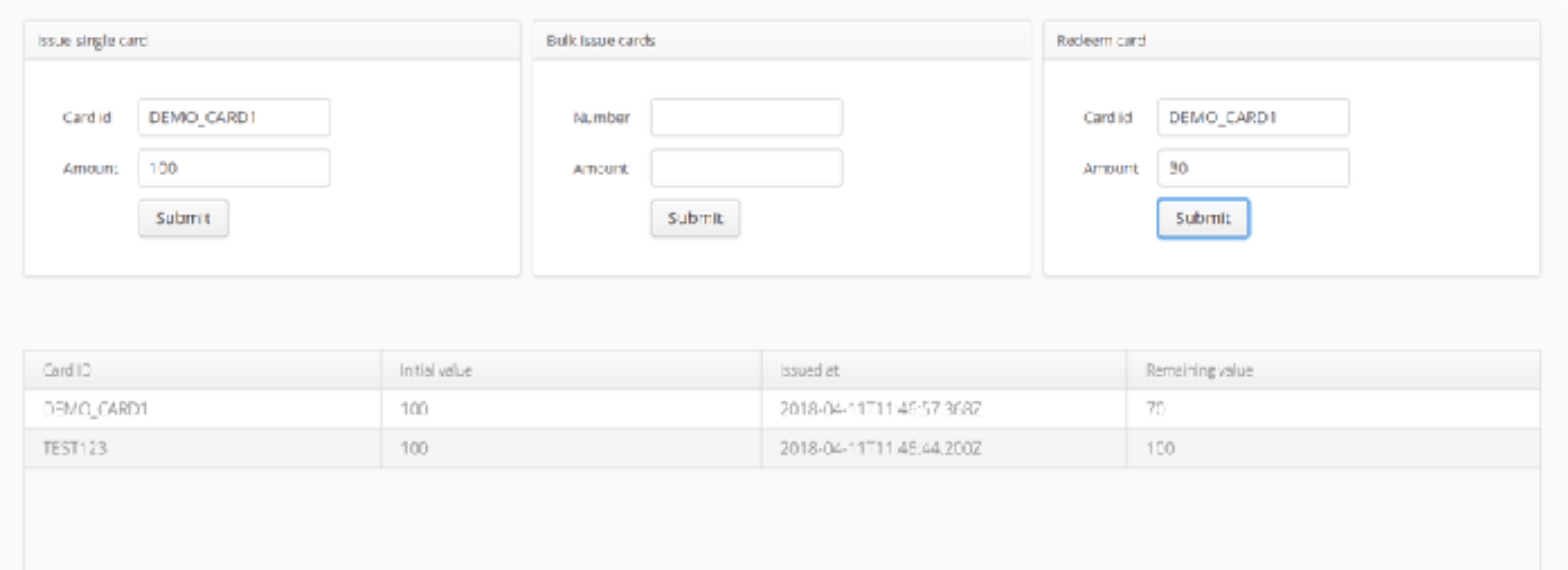


Install yourself, use Minikube, managed through GCP, AWS or Azure, or use Pivotal's PKS

Super Simple Sample System: Gift Cards

- Gift cards get **issued** at a certain initial value.

- Each time they are used to buy something **("redeemed")** the remaining value is adjusted.



Issue single card

Card id	DEMO_CARD1
Amount	100

Bulk issue cards

Number	
Amount	

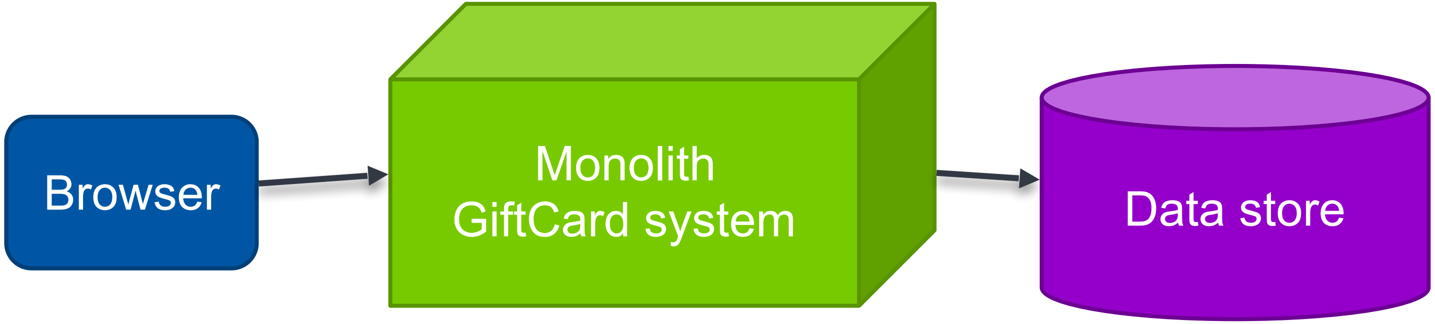
Redeem card

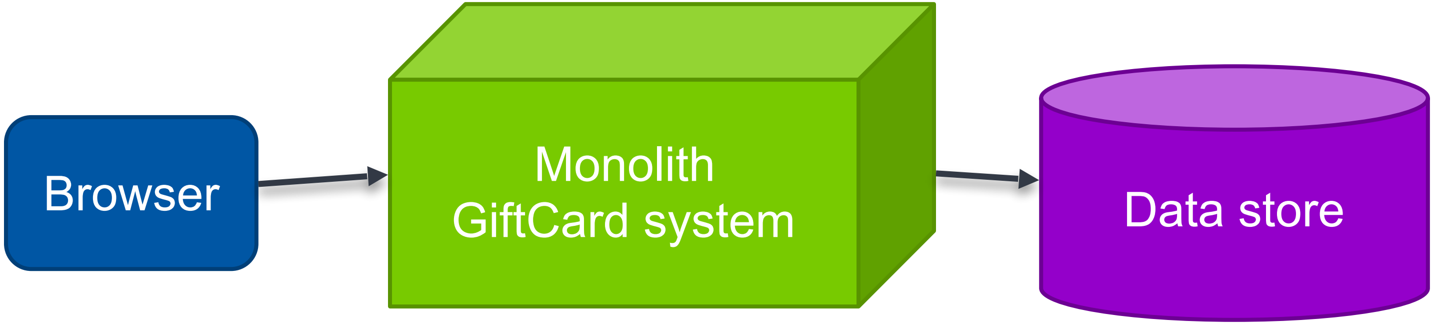
Card id	DEMO_CARD1
Amount	30

Card ID	Initial value	Issued at	Remaining value
DEMO_CARD1	100	2018-04-11T11:46:57.368Z	70
TEST123	100	2018-04-11T11:45:44.200Z	100

https://github.com/AxonIQ/giftcard-demo-series

AxonIQ


Browser
Monolith
GiftCard system
Data store



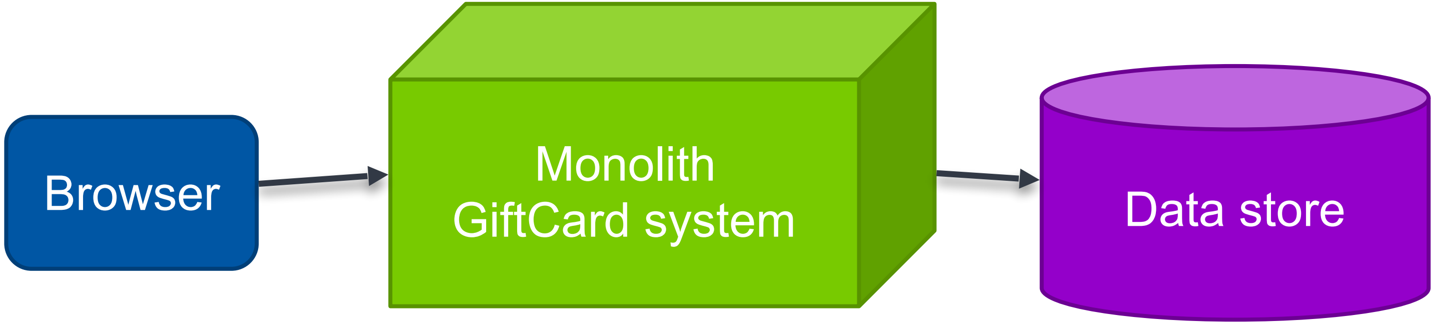
How to transform our monolith into a agile, scalable, microservices system running on Kubernetes?

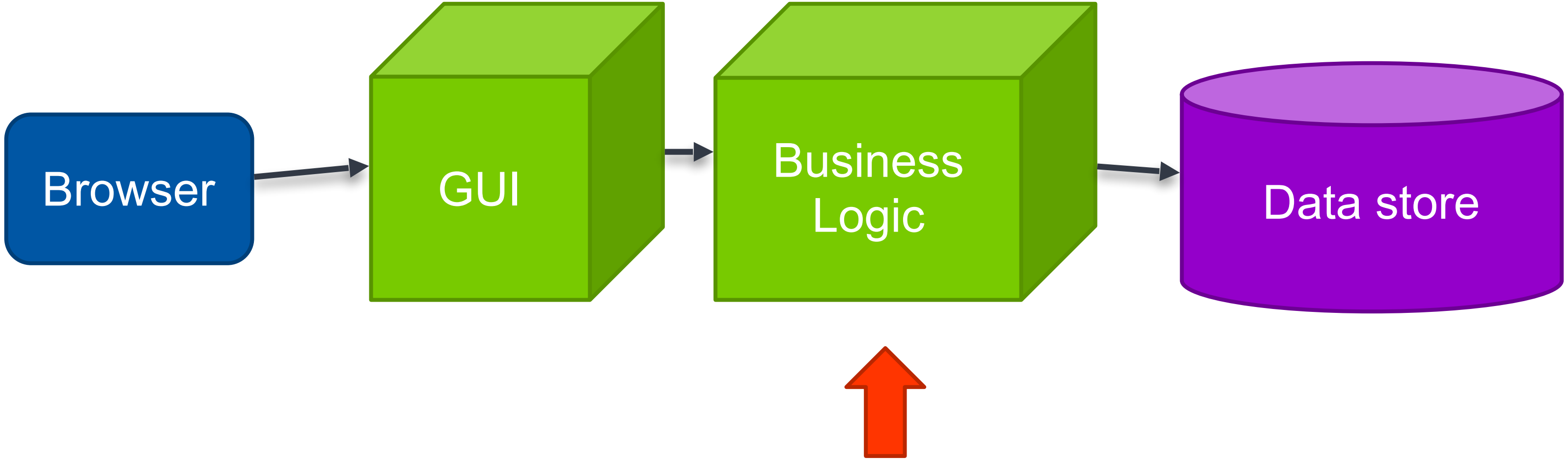
1. Decide on component split.

2. Requirements for messaging between the components?

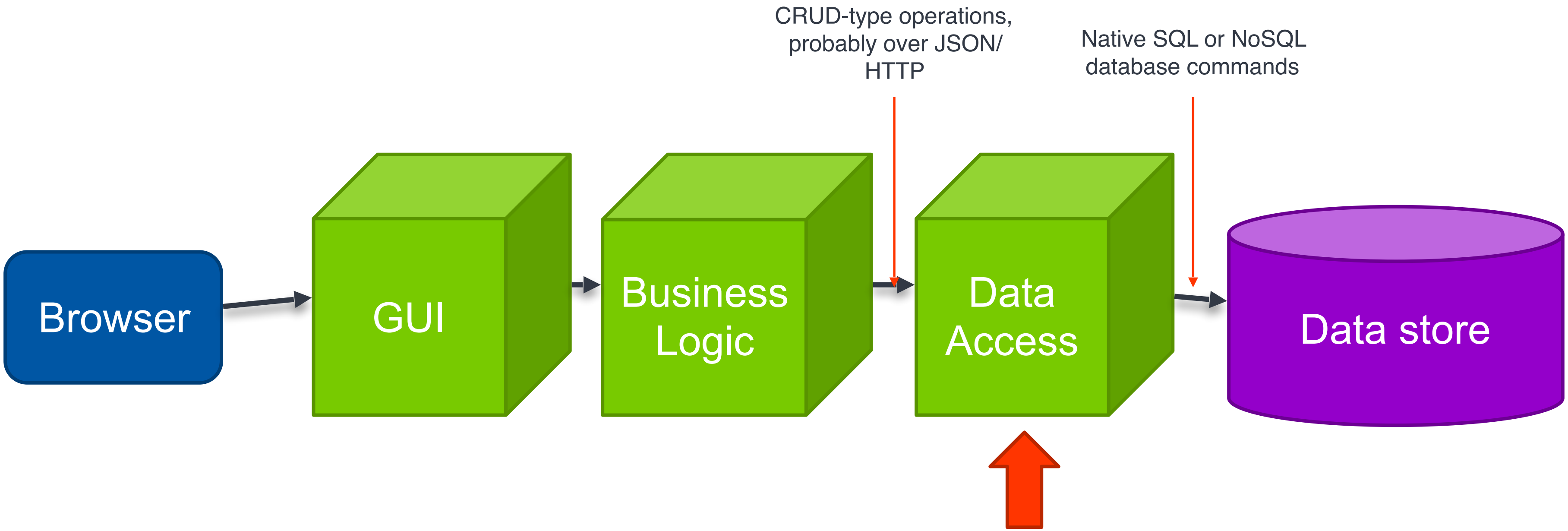
3. Practical implementation

with Axon Framework, AxonHub, Spring Boot, Docker, Kubernetes, Google Cloud Platform

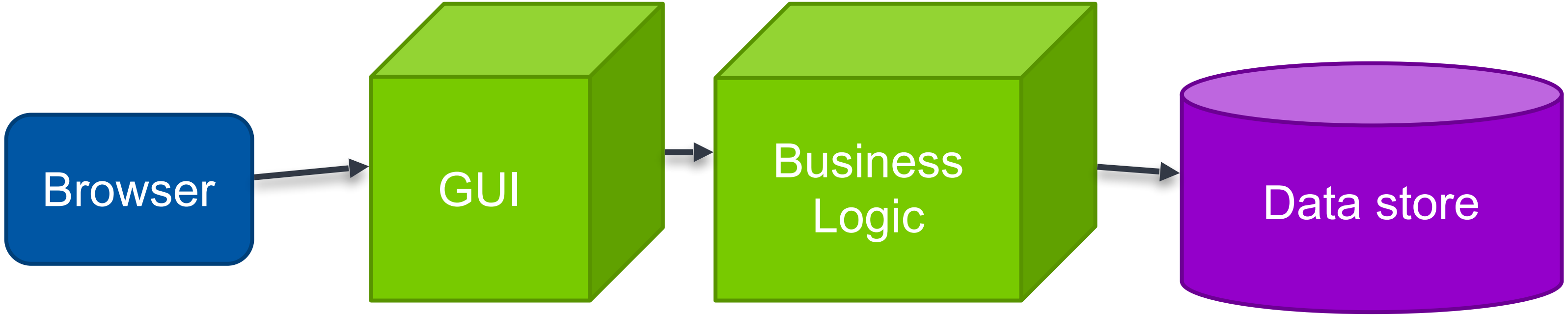

Browser
Monolith
GiftCard system
Data store

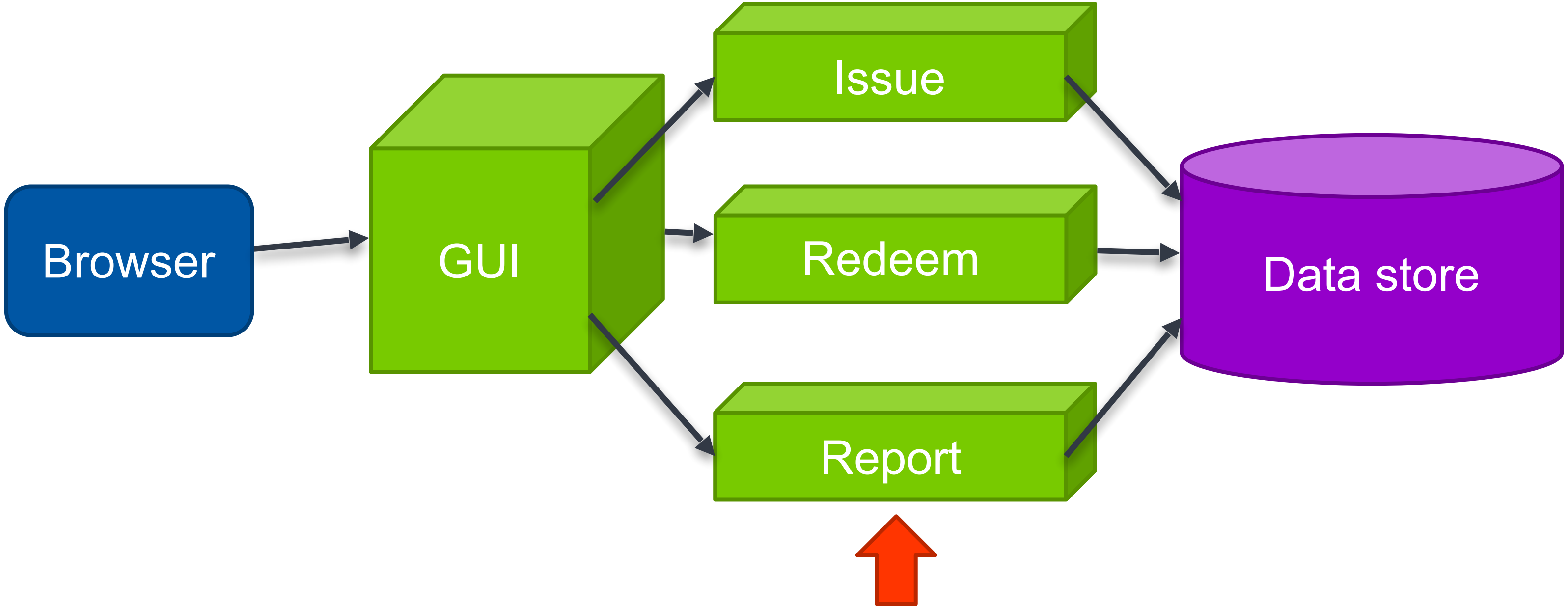


This is still a very broad functionality (as the system grows). Not micro.


CRUD-type operations, probably over JSON/ HTTP
Native SQL or NoSQL database commands
Browser
GUI
Business Logic
Data Access
Data store

This doesn't offer clear advantages.
It has very clear disadvantages.


Browser
GUI
Business Logic
Data store



Actual agility and scalability will be limited by having the single data store.

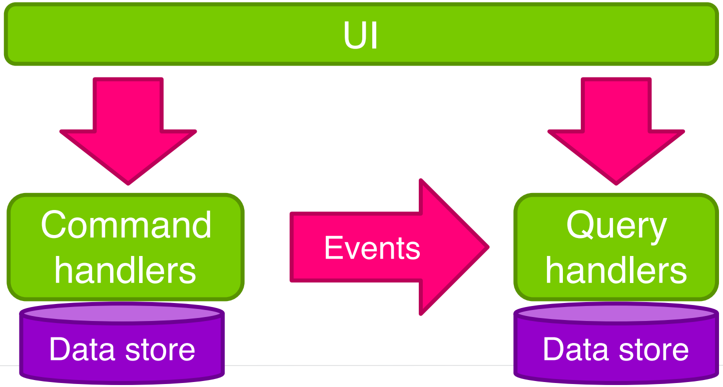
Let's look at 2 pretty old ideas

… that are really useful in this context.

Command-Query Responsibility Segregation (CQRS)

Idea

- Separate datastores for processing commands (changing state) vs. queries (providing info).



Advantages

- Query simplicity
- Optimal technology choice
- Easily scalable query side.
- Less workload on command side.

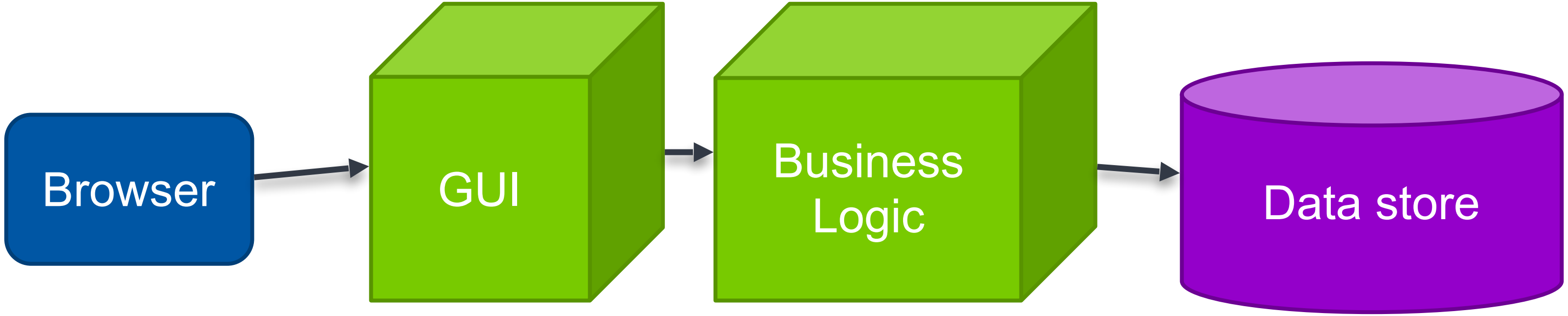
Domain-Driven Design (DDD) and Aggregates

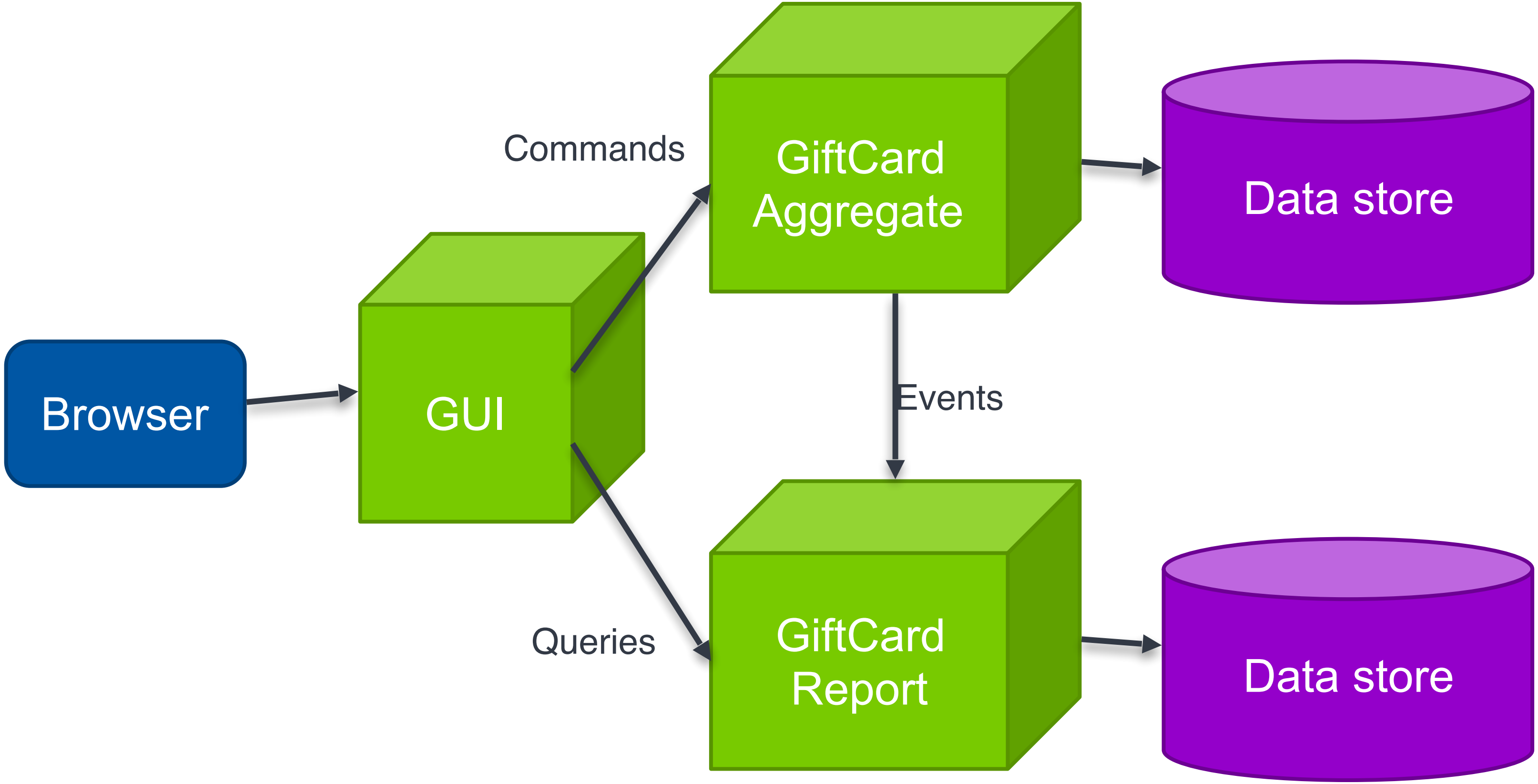
Idea

- Use DDD to align your system architecture with the domain.
- Use DDD's concept of "Aggregate" to create consistency boundaries inside a model.

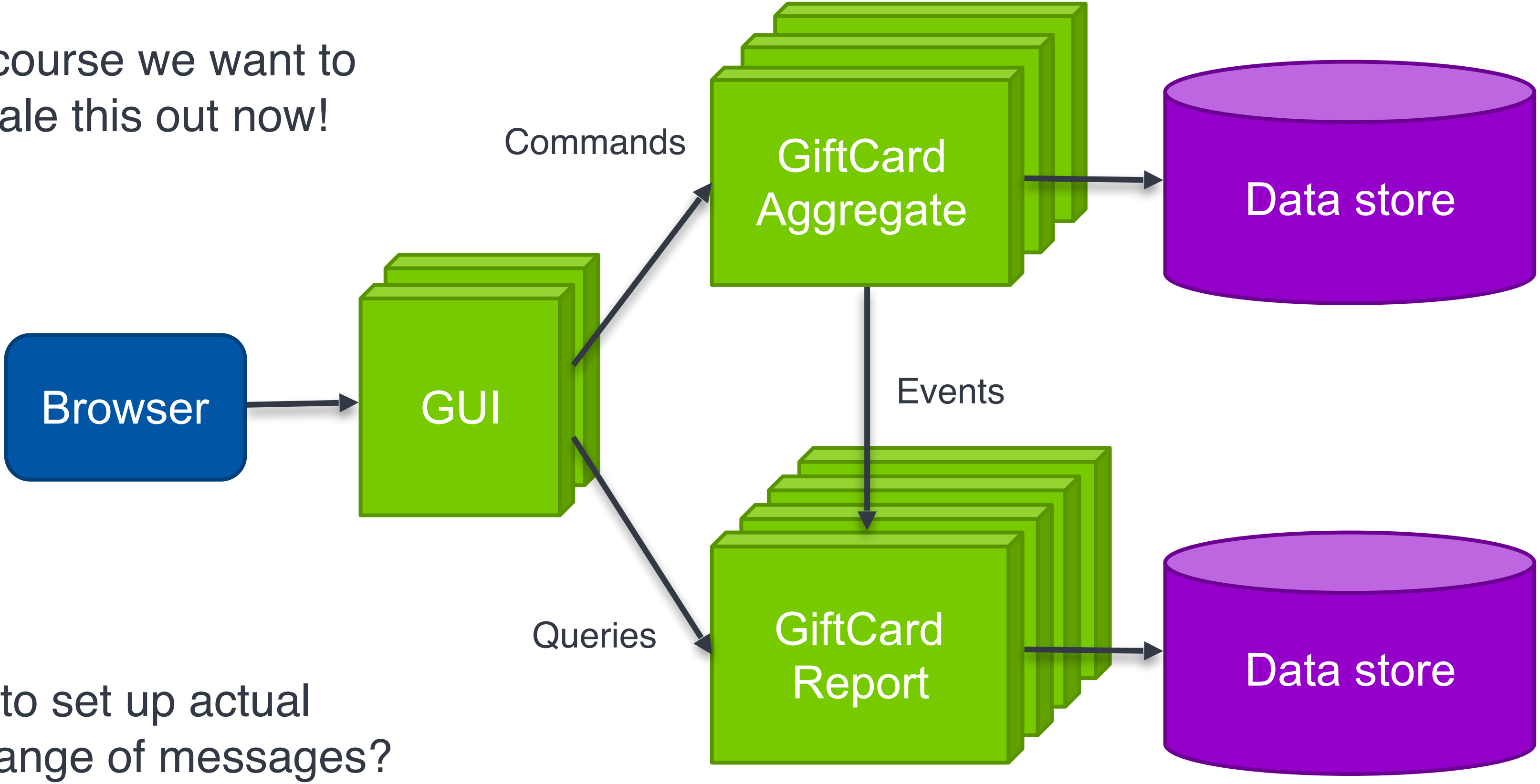
Advantages

- Broadly: having your microservices align with a domain model helps make them independently evolvable.
- Aggregates help size command-side microservices:
  - 2 aggregates should never "have to be" in the same service
  - 1 aggregate should not be "spread across" multiple services


Browser
GUI
Business Logic
Data store


Commands
GiftCard Aggregate
Data store
Browser
GUI
Events
Queries
GiftCard Report
Data store

Of course we want to scale this out now!



How to set up actual exchange of messages?

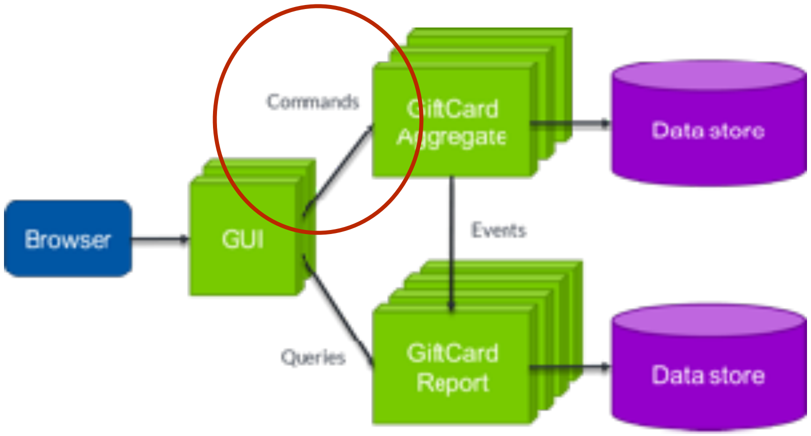
Message requirements

Generic requirements to make this work at scale:

- Fast, efficient, asynchronous
- Some form of load balancing
- Dynamic scaling of nodes

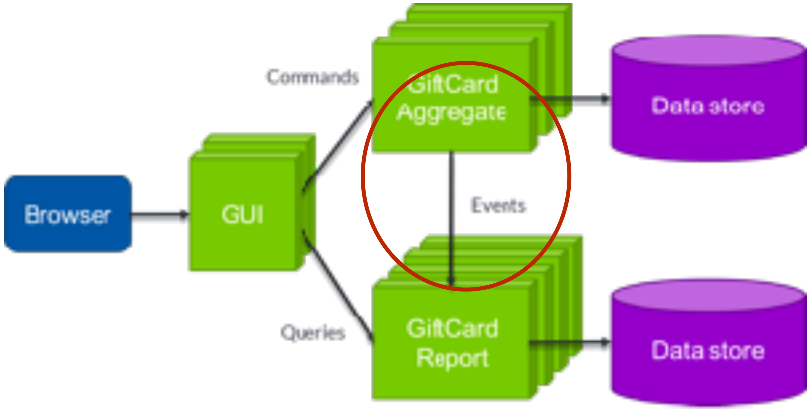
There are also a number of specific requirements for the 3 message stereotypes command, event and query!

Command messages



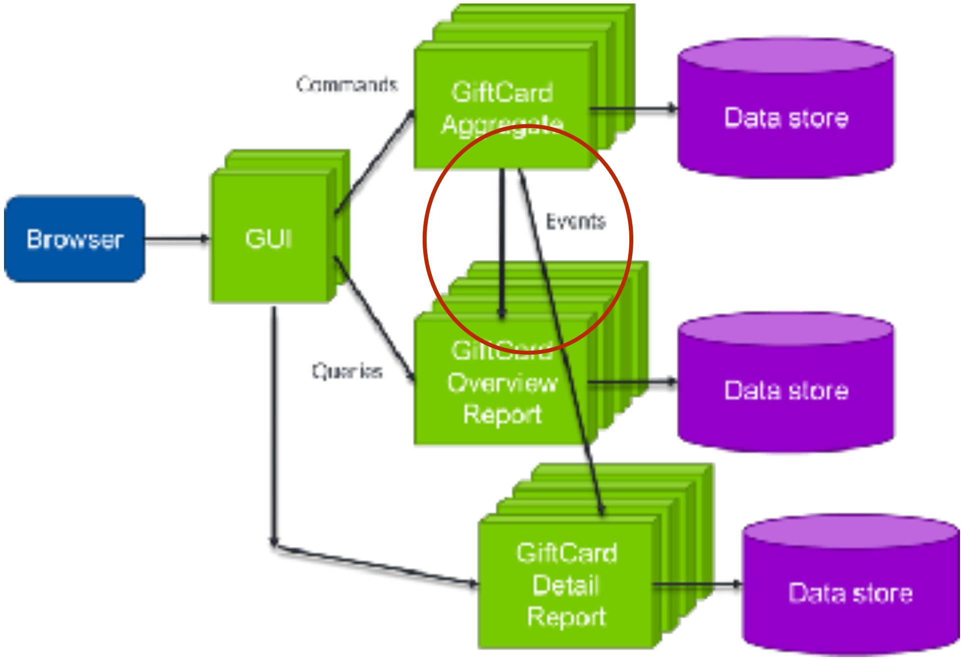
- Route to a single handler instance (load balancing)
- Associated response required by client (success/failure).
- Use consistent routing based on aggregate id

Event messages



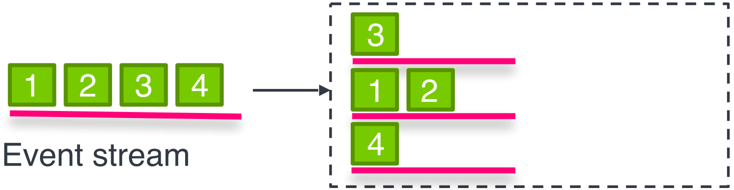
- In this case, each event should lead to 1 update on data store (competing consumer).

Event messages



- The general rule: handled once by every logical event handler.

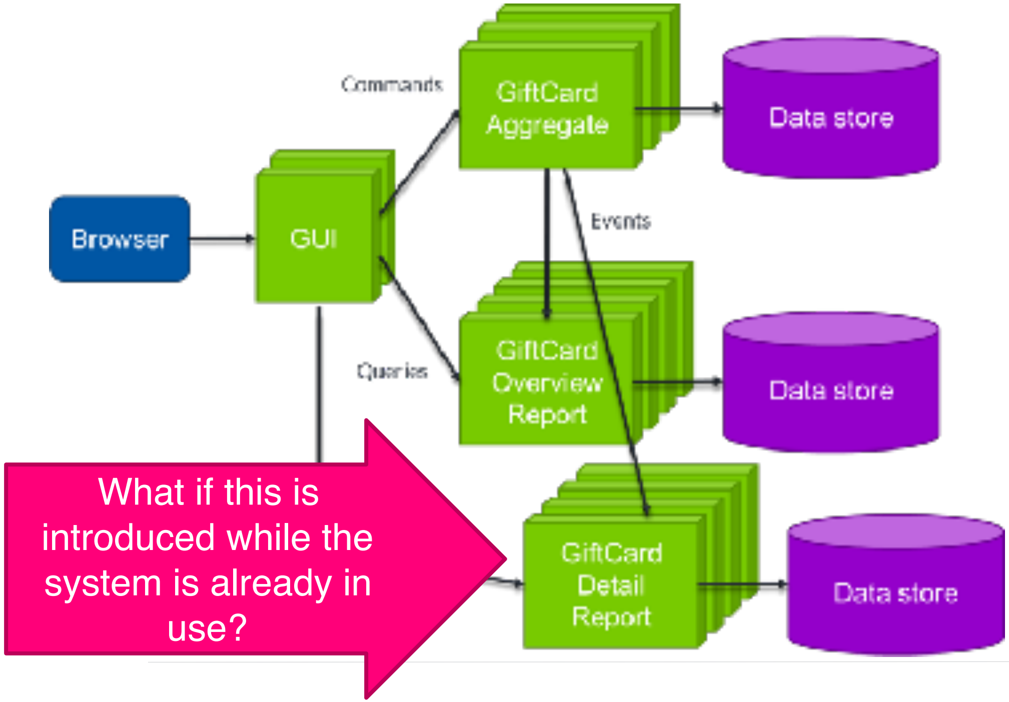
Event messages



- The general rule: handled once by every logical event handler.
- Parallel processing is desirable, but should follow a sequencing policy.

- "3" may get processed before "2"
- If "2" is "Create Card X" and "3" is "Redeem Card X" this may fail

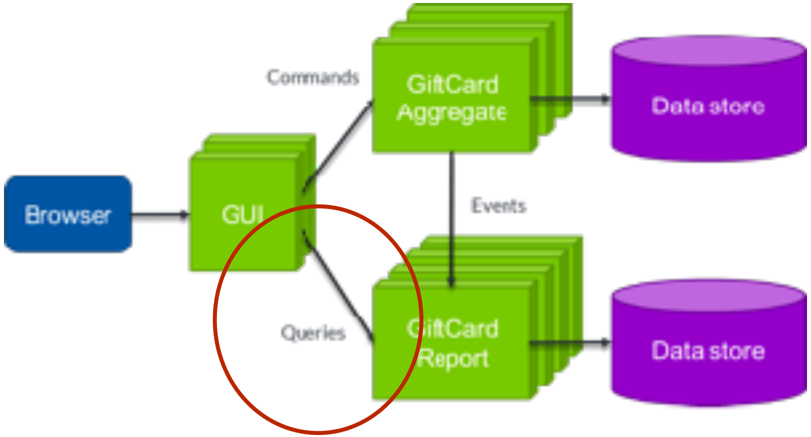
Event messages



What if this is introduced while the system is already in use?

- The general rule: handled once by every logical event handler.
- Parallel processing is desirable, but should follow a sequencing policy.
- Event *replay* is a very convenient feature to initialize new read models.

Query messages



- Route to a single handler instance (load balancing)
- Associated response required by client.
- More advanced query patterns could also occur (scatter-gather).

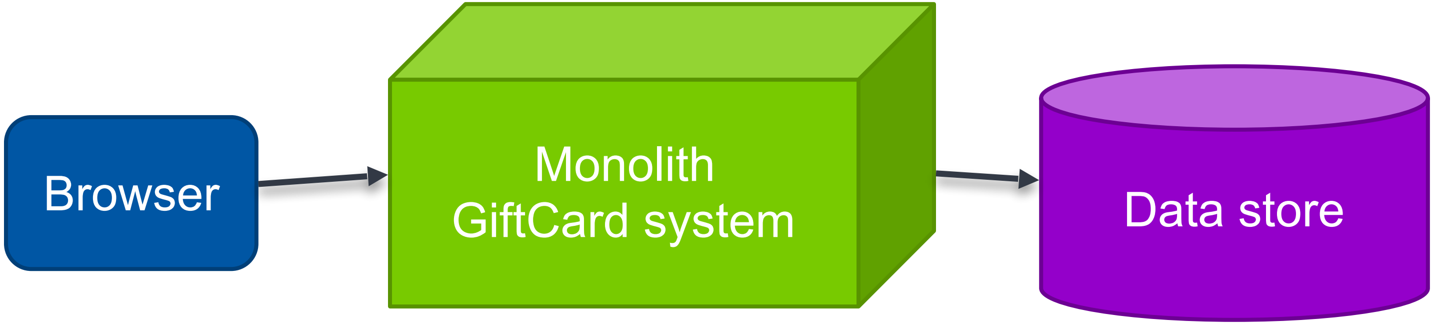
Message requirements

Generic

- Fast, efficient, asynchronous
- Some form of load balancing
- Dynamic scaling of nodes

Stereotype-specific

- **Commands:** single execution consistent routing, responses
- **Events:** processed once per logical handler, sequencing policy, no responses, replay
- **Queries:** load balancing, responses, usually (not always) executed once.



How to transform our monolith into a scalable, agile microservices system running on Kubernetes?

✓ 1. How to break it up?

✓ 2. Requirements for messaging between the components?

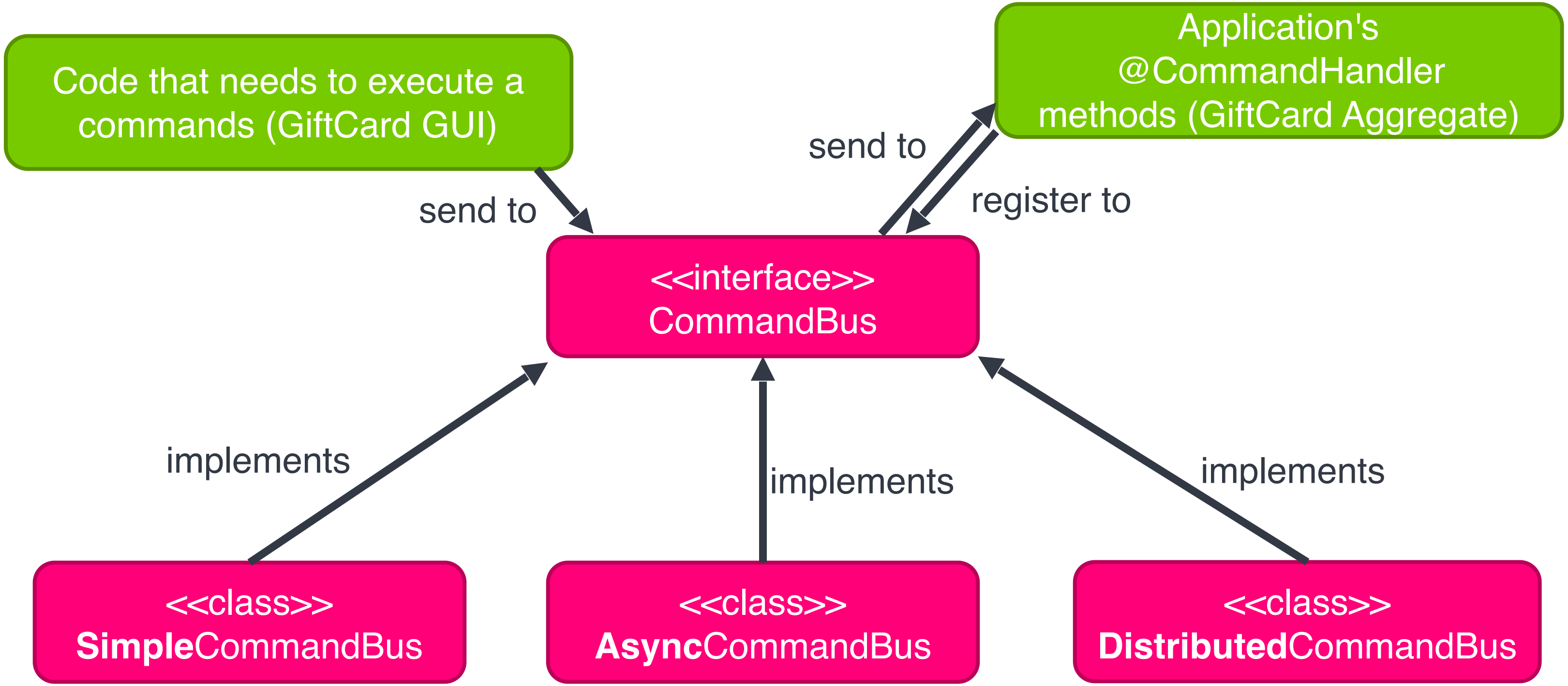
3. Practical implementation
with Axon Framework, AxonHub, Spring Boot, Docker, Kubernetes, Google Cloud Platform

Axon Framework

- Open source Java Framework.
- Started 7 years ago.
- 800k+ downloads, 50% of which in the past 6 months.
- Started out as "CQRS Framework", currently used a lot for microservices.
- Key principle: **location transparency**

Location transparency in Axon

Commands as illustration – Queries and Events work similarly



Code that needs to execute a commands (GiftCard GUI)

send to

Application's @CommandHandler methods (GiftCard Aggregate)

send to

register to

<<interface>>
CommandBus

implements

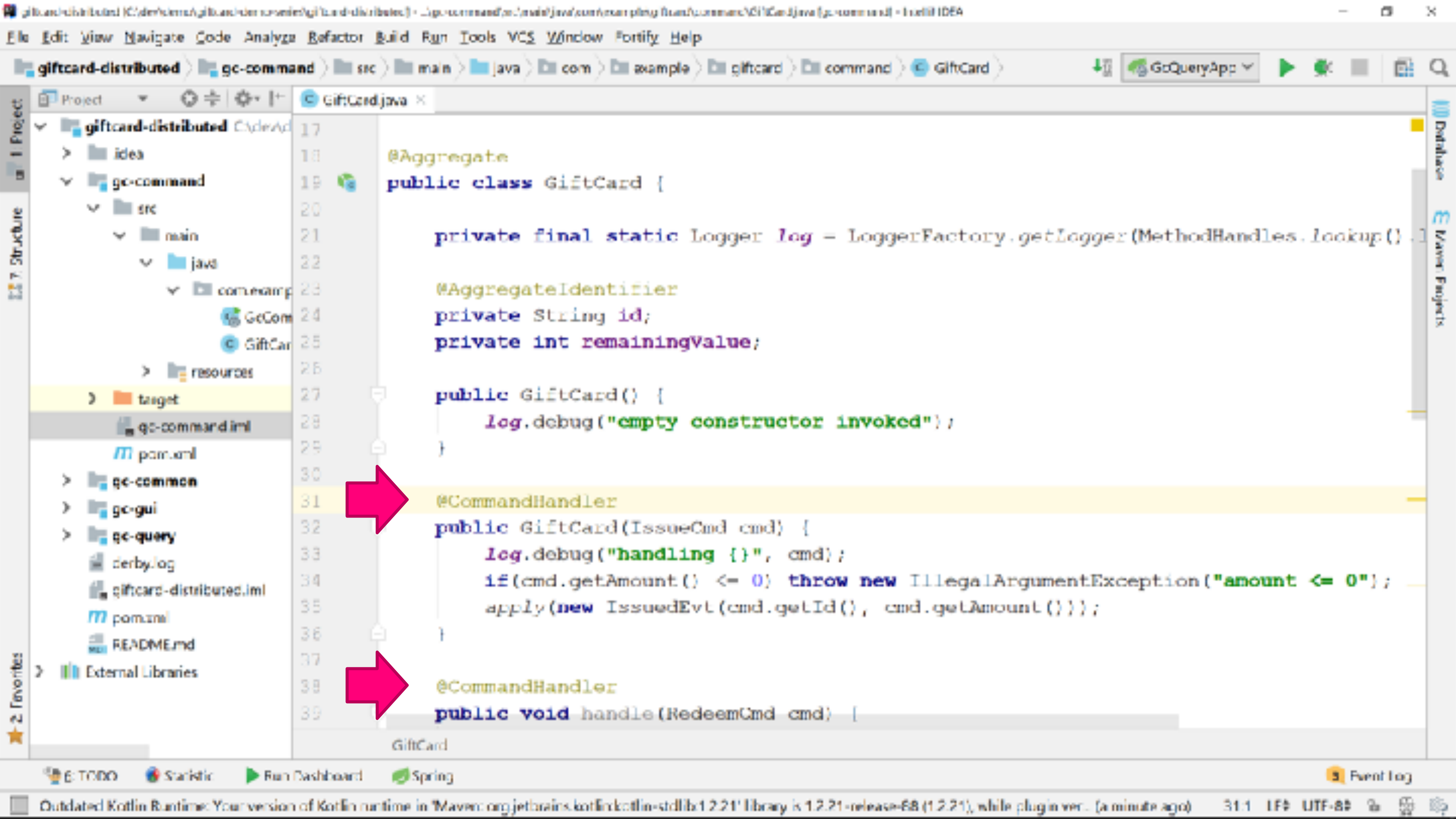
implements

implements

<<class>>
SimpleCommandBus

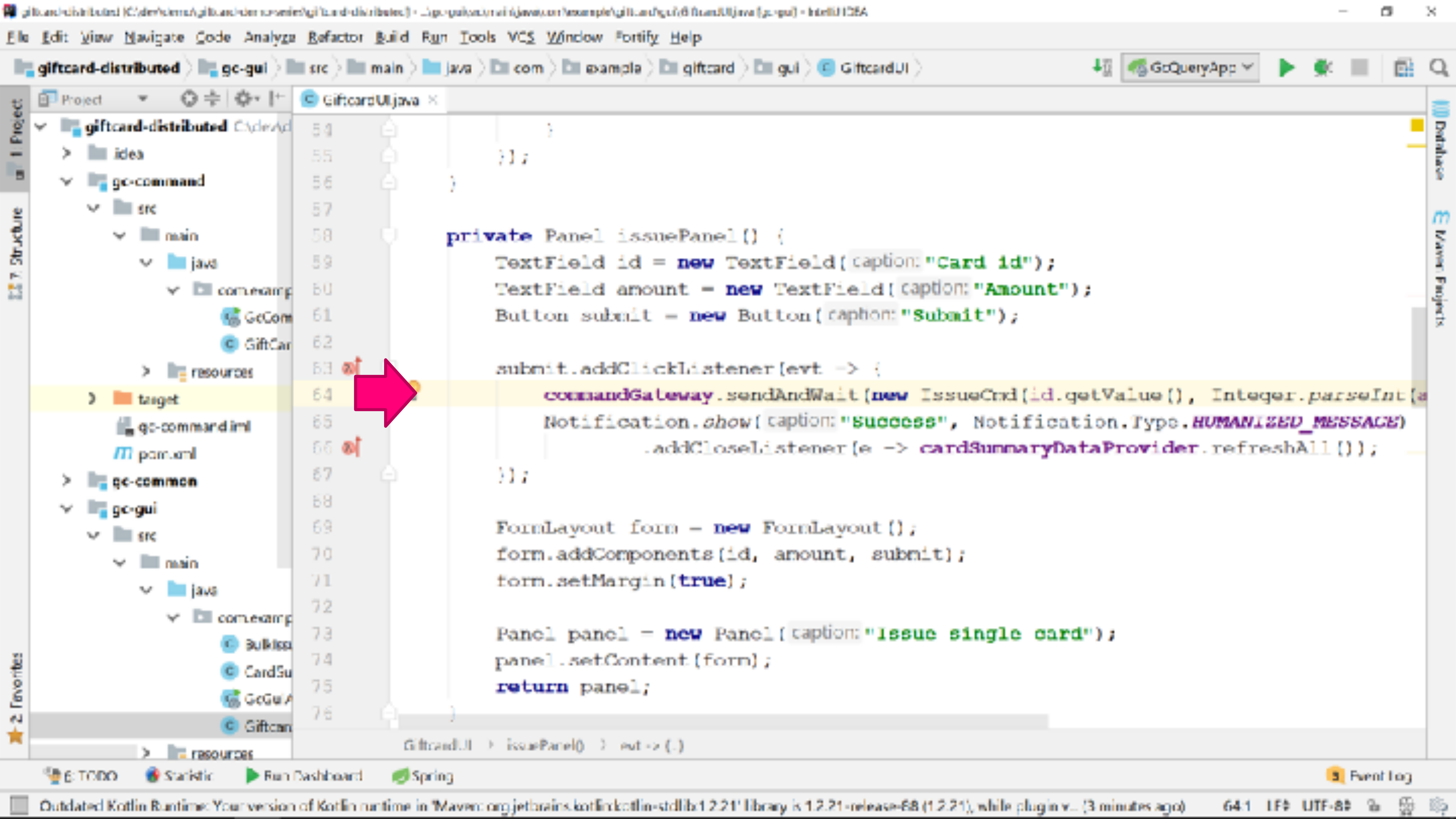
<<class>>
AsyncCommandBus

<<class>>
DistributedCommandBus



```
@Aggregate
public class GiftCard {

    private final static Logger log = LoggerFactory.getLogger(MethodHandles.lookup().l

    @AggregateIdentifier
    private String id;
    private int remainingValue;

    public GiftCard() {
        log.debug("empty constructor invoked");
    }

    @CommandHandler
    public GiftCard(IssueCmd cmd) {
        log.debug("handling {}", cmd);
        if(cmd.getAmount() <= 0) throw new IllegalArgumentException("amount <= 0");
        apply(new IssuedEvt(cmd.getId(), cmd.getAmount()));
    }

    @CommandHandler
    public void handle(RedeemCmd cmd) {
```

Outdated Kotlin Runtime: Your version of Kotlin runtime in 'Maven: org.jetbrains.kotlin:kotlin-stdlib:1.2.21' library is 1.2.21-release-68 (1.2.21), while plugin ver... (a minute ago)



```java
                }
            });
        }

    private Panel issuePanel() {
        TextField id = new TextField( caption: "Card id");
        TextField amount = new TextField( caption: "Amount");
        Button submit = new Button( caption: "Submit");

        submit.addClickListener(evt -> {
            commandGateway.sendAndWait(new IssueCmd(id.getValue(), Integer.parseInt(a
            Notification.show( caption: "Success", Notification.Type.HUMANIZED_MESSAGE)
                    .addCloseListener(e -> cardSummaryDataProvider.refreshAll());
        });

        FormLayout form = new FormLayout();
        form.addComponents(id, amount, submit);
        form.setMargin(true);

        Panel panel = new Panel( caption: "Issue single card");
        panel.setContent(form);
        return panel;
    }
```

Outdated Kotlin Runtime: Your version of Kotlin runtime in 'Maven: org.jetbrains.kotlin:kotlin-stdlib:1.2.21' library is 1.2.21-release-68 (1.2.21), while plugin v... (3 minutes ago)

Practical distribution with Axon

Until recently

Commands

SpringCloud + HTTP
or
JGroups

Events

Either AMQP
or
tracking a database table

Queries

No standard functionality; typically custom REST interface with HTTP load balancing.

Complex to set up correctly.
Doesn't meet all requirements.

Practical distribution with Axon

Commands

SpringCloud + HTTP
or
JGroups

or
AxonHubCommandBus

Events

Either AMQP
or
tracking a database table

or
AxonHubEventBus

Queries

No standard functionality; typically custom REST interface with HTTP load balancing

or
AxonHubQueryBus

AxonHub essentials

- Server system, together with open source client/driver.
- Unified messaging for all 3 stereotypes
- Intelligent: near-zero configuration

How does it work?

When an application has an AxonHub*Bus, two things happen:

1. When that application puts a message on the bus, it will be sent to AxonHub for further processing.

2. The application will actively inform AxonHub about the @CommandHandlers, @QueryHandlers and @EventHandlers it has.

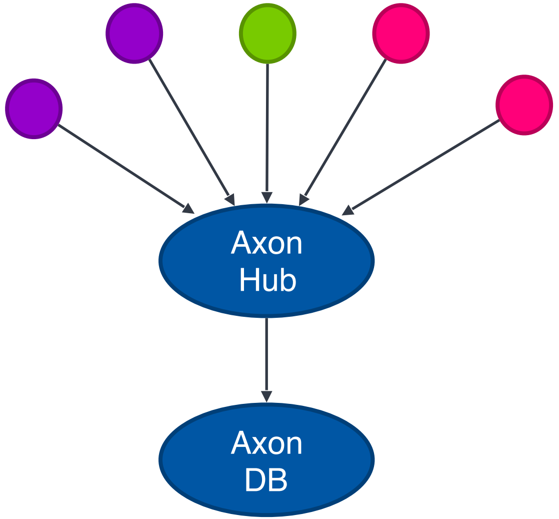
In combination with

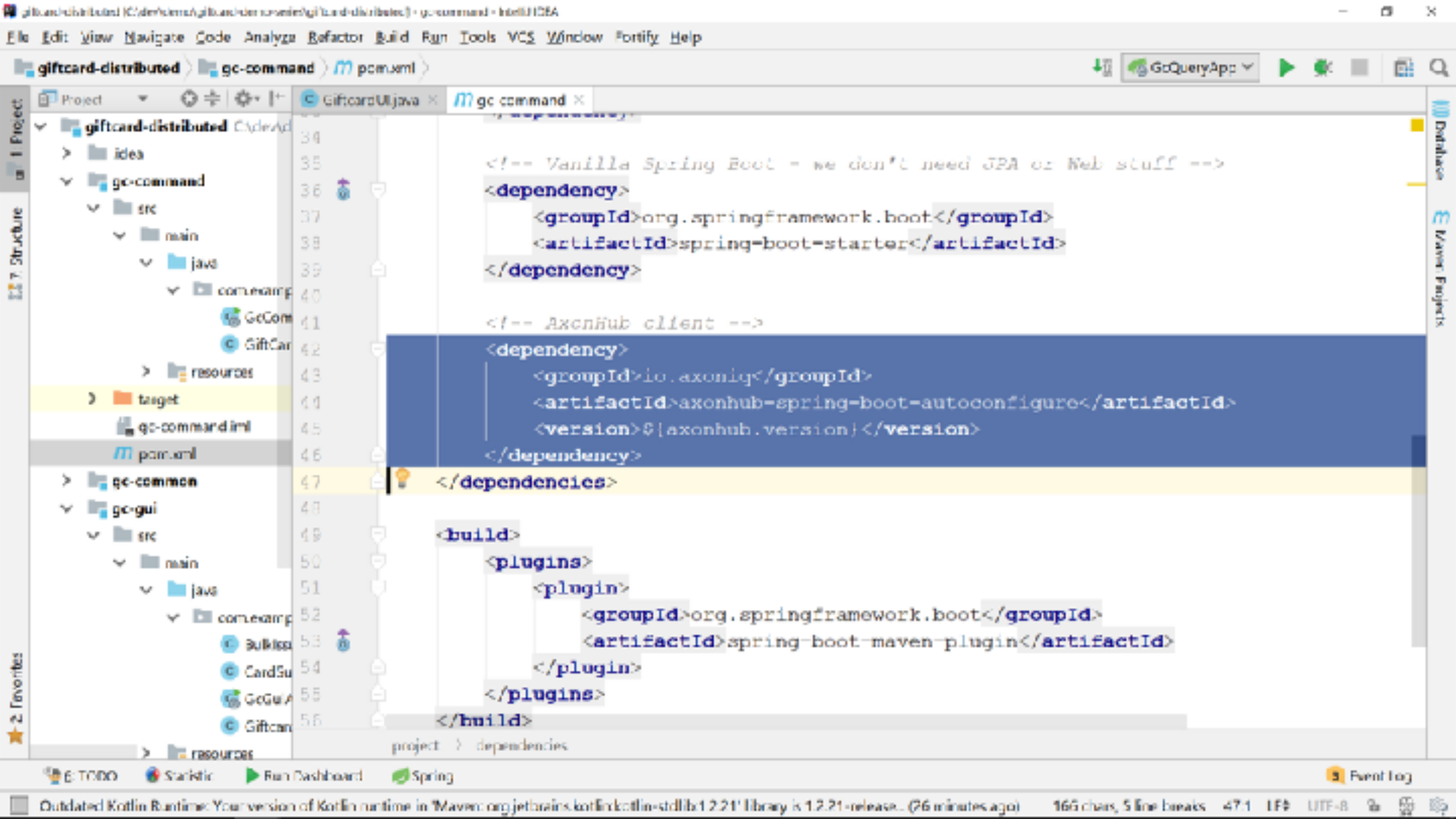
- AxonHub's understanding of message meta-data and routing patterns,
- 2-way gRPC connections

this allows fully automatic routing and monitoring / management functionality.

AxonHub and AxonDB

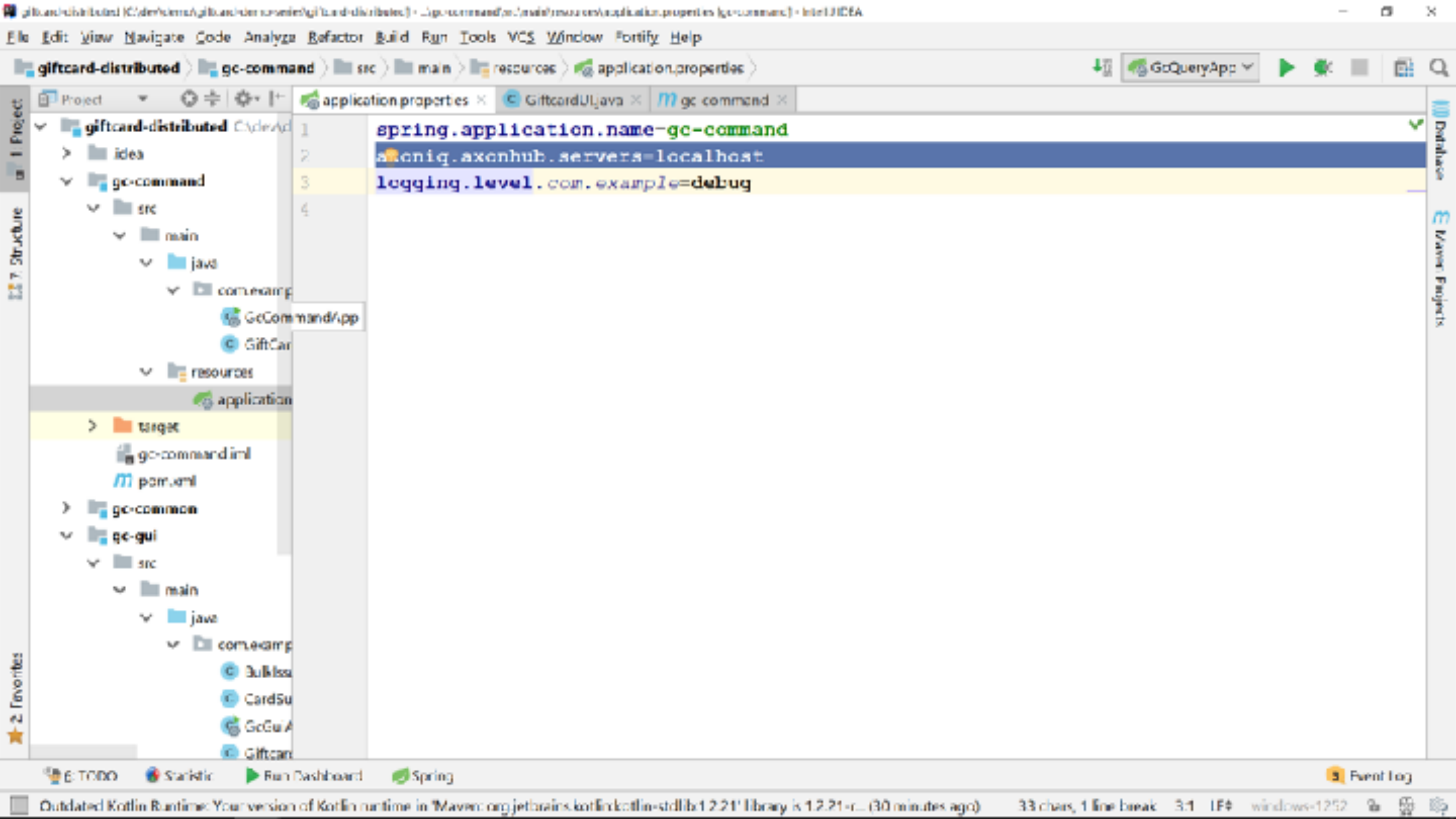
- Events go through AxonHub doesn't store events itself – it needs an event store for that.
- AxonDB is AxonIQ built-for-purpose event store.





```
        </dependency>

        <!-- Vanilla Spring Boot - we don't need JPA or Web stuff -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter</artifactId>
        </dependency>

        <!-- AxonHub client -->
        <dependency>
            <groupId>io.axoniq</groupId>
            <artifactId>axonhub-spring-boot-autoconfigure</artifactId>
            <version>${axonhub.version}</version>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
```

Outdated Kotlin Runtime: Your version of Kotlin runtime in 'Maven: org.jetbrains.kotlin:kotlin-stdlib:1.2.21' library is 1.2.21-release... (26 minutes ago)   166 chars, 5 line breaks   47:1   LF   UTF-8



```
spring.application.name=gc-command
axoniq.axonhub.servers=localhost
logging.level.com.example=debug
```

Outdated Kotlin Runtime: Your version of Kotlin runtime in 'Maven: org.jetbrains.kotlin:kotlin-stdlib:1.2.21' library is 1.2.21-r... (30 minutes ago)   33 chars, 1 line break   3:1   LF‡   windows-1252

Kubernetes deployment

- gc-command
- gc-query
- gc-gui

All 3 are standard Kubernetes "deployments" of a Pod with a single Docker container. Pods are ephemeral.

- axonhub
- axondb

Each need to operate as a cluster for HA
Won't work as a "deployments"
Instead, using "StatefulSet".

- Pods get **identity** (network, storage)
- Scaled up and down sequentially

Deploying gc-command

Dockerfile

```
FROM openjdk:8-jdk-alpine
VOLUME /tmp
ADD gc-command-1.0.jar app.jar
ADD application.properties .
ENTRYPOINT ["java","-jar","/app.jar"]
```

application.properties

```
axoniq.axonhub.servers=axonhub.default.svc.cluster.local
```

When creating the Kubernetes StatefulSet, this DNS SRV entry will be created and point to all nodes of the cluster.

Deploying gc-command

Pushing this to Google Kubernetes Engine via Google Container Repository:

```
docker build -t gc-command .
docker tag gc-command eu.gcr.io/giftcard-distributed/gc-command
docker push eu.gcr.io/giftcard-distributed/gc-command

kubectl run gc-command --image eu.gcr.io/giftcard-distributed/gc-
command --replicas=1
```

axonhub.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: axonhub
  labels:
    app: axonhub
spec:
  ports:
  - port: 8124
    name: grpc
    targetPort: 8124
  clusterIP: None
  selector:
    app: axonhub
---
```

AxonHub gRPC port: headless service, all instances registered in DNS by Kubernetes

```
apiVersion: v1
kind: Service
metadata:
  name: axonhub-gui
  labels:
    app: axonhub
spec:
  ports:
  - port: 8024
    name: gui
    targetPort: 8024
  selector:
    app: axonhub
  type: LoadBalancer
---
```

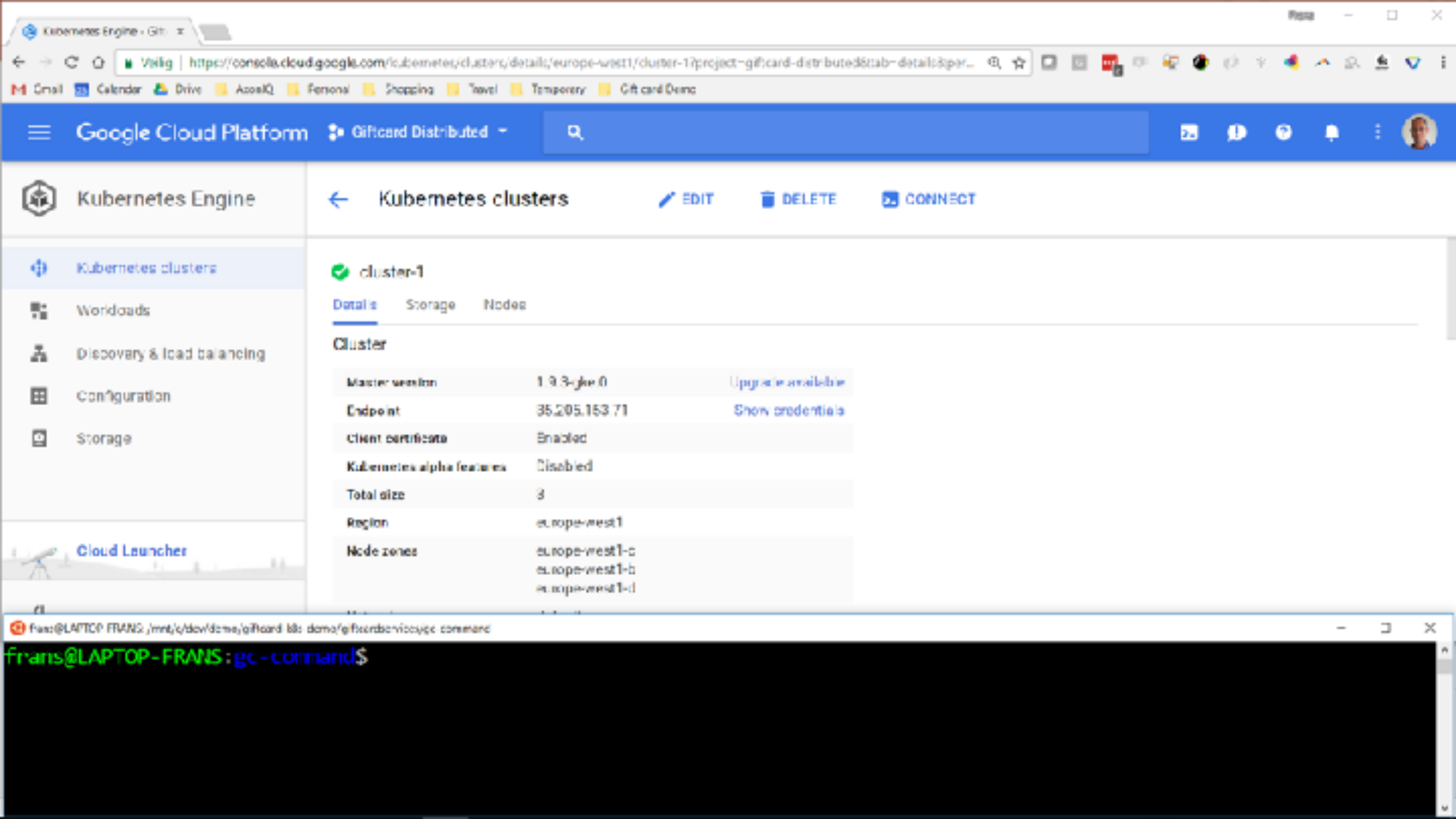
AxonHub management GUI: behind HTTP load balancer

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: axonhub
spec:
  serviceName: "axonhub"
  replicas: 3
  selector:
    matchLabels:
      app: axonhub
  template:
    metadata:
      labels:
        app: axonhub
    spec:
      containers:
      - name: axonhub
        image: eu.gcr.io…
        ports:
        - containerPort: 8124
          protocol: TCP
          name: grpc
        - containerPort: 8024
          protocol: TCP
          name: gui
        readinessProbe:
          …
```

AxonIQ

Switching to live demo now

(but there are some slides after this one as backup)



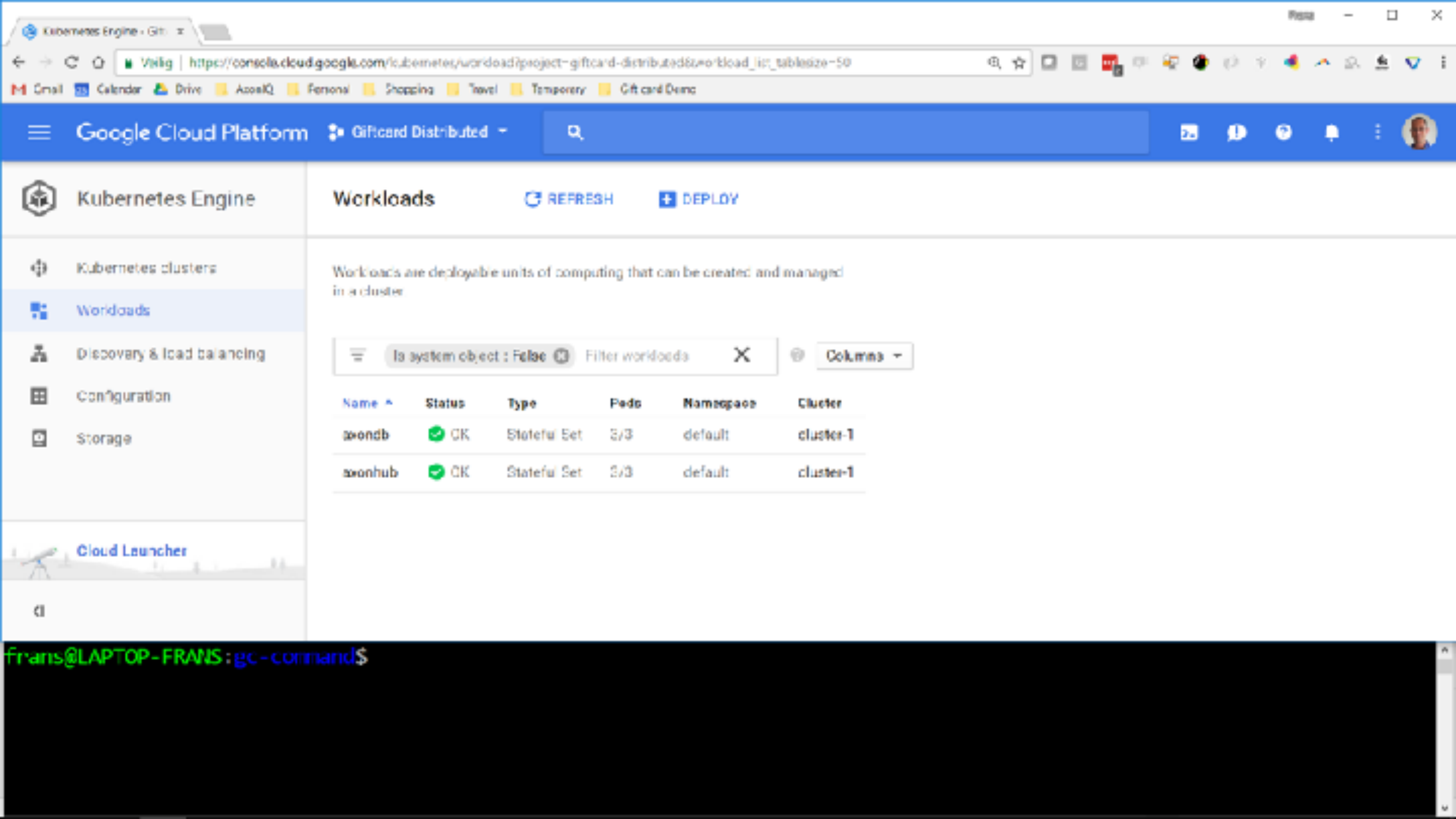
Kubernetes clusters

cluster-1

Cluster

Master version	1.9.3-gke.0	Upgrade available
Endpoint	35.205.153.71	Show credentials
Client certificate	Enabled	
Kubernetes alpha features	Disabled	
Total size	3	
Region	europe-west1	
Node zones	europe-west1-c europe-west1-b europe-west1-d	

```
frans@LAPTOP-FRANS: /mnt/c/dev/demo/giftcard-k8s-demo/giftcardservices/gc-command
frans@LAPTOP-FRANS:gc-command$
```

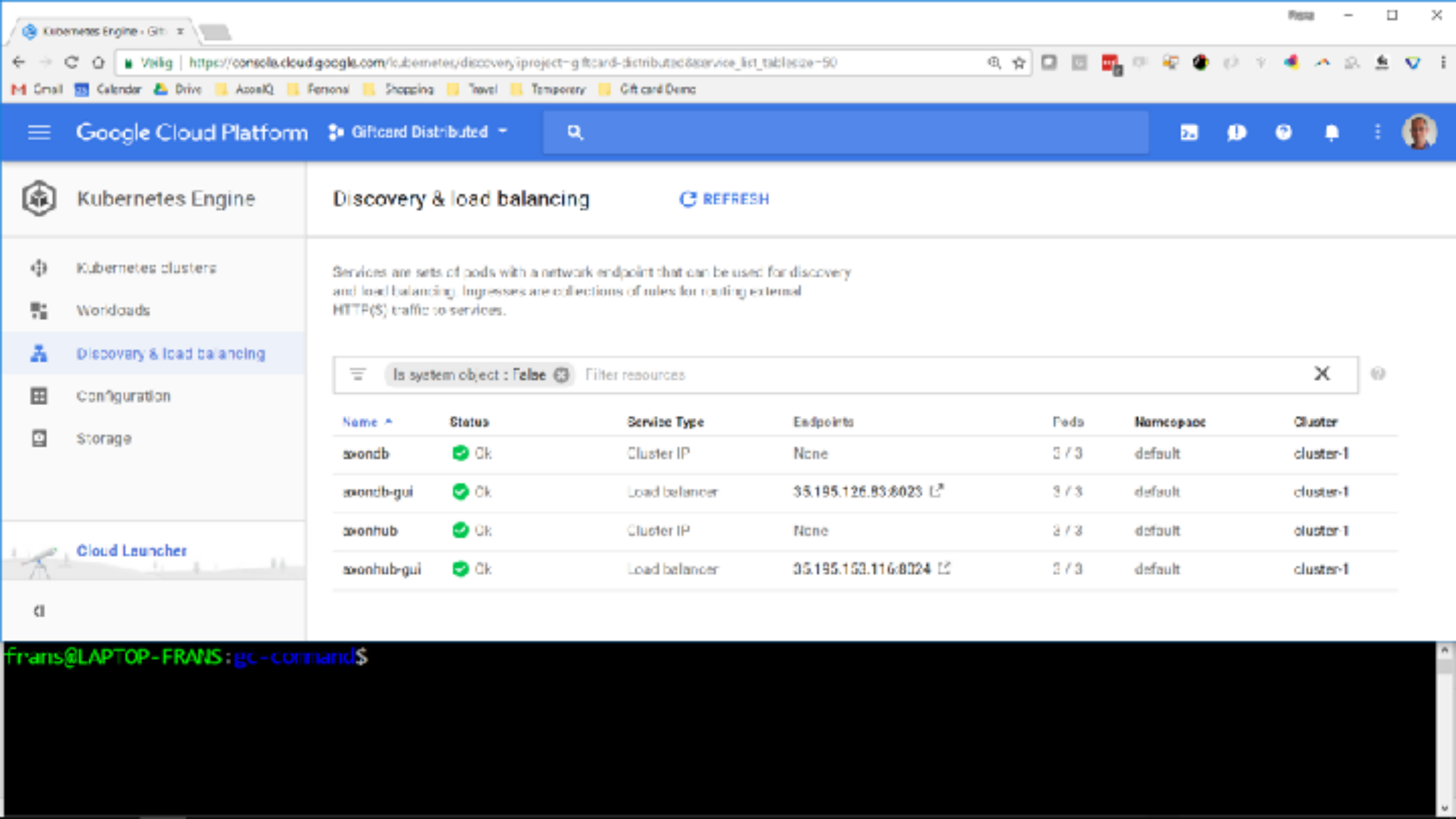


Workloads

Workloads are deployable units of computing that can be created and managed in a cluster

Name ^	Status	Type	Pods	Namespace	Cluster
axondb	✓ OK	Stateful Set	3/3	default	cluster-1
axonhub	✓ OK	Stateful Set	3/3	default	cluster-1

```
frans@LAPTOP-FRANS:gc-command$
```

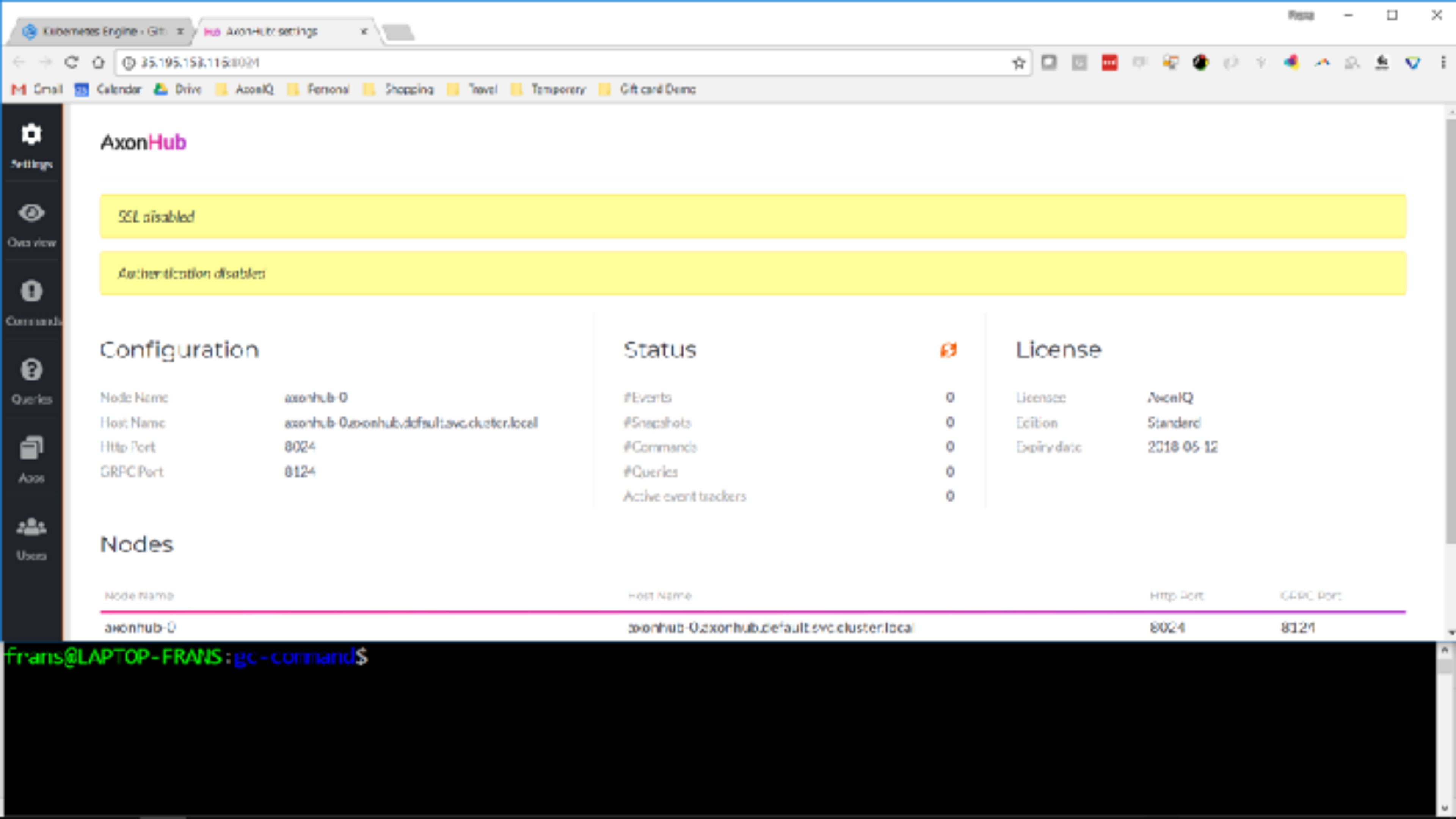


Discovery & load balancing

Services are sets of pods with a network endpoint that can be used for discovery and load balancing. Ingresses are collections of rules for routing external HTTP(S) traffic to services.

Name ▲	Status	Service Type	Endpoints	Pods	Namespace	Cluster
axondb	✓ Ok	Cluster IP	None	3 / 3	default	cluster-1
axondb-gui	✓ Ok	Load balancer	35.195.126.83:8023	3 / 3	default	cluster-1
axonhub	✓ Ok	Cluster IP	None	3 / 3	default	cluster-1
axonhub-gui	✓ Ok	Load balancer	35.195.153.116:8024	3 / 3	default	cluster-1

```
frans@LAPTOP-FRANS:gc-command$
```



AxonHub

SSL disabled

Authentication disabled

Configuration

Node Name	axonhub-0
Host Name	axonhub-0.axonhub.default.svc.cluster.local
Http Port	8024
GRPC Port	8124

Status

#Events	0
#Snapshots	0
#Commands	0
#Queries	0
Active event trackers	0

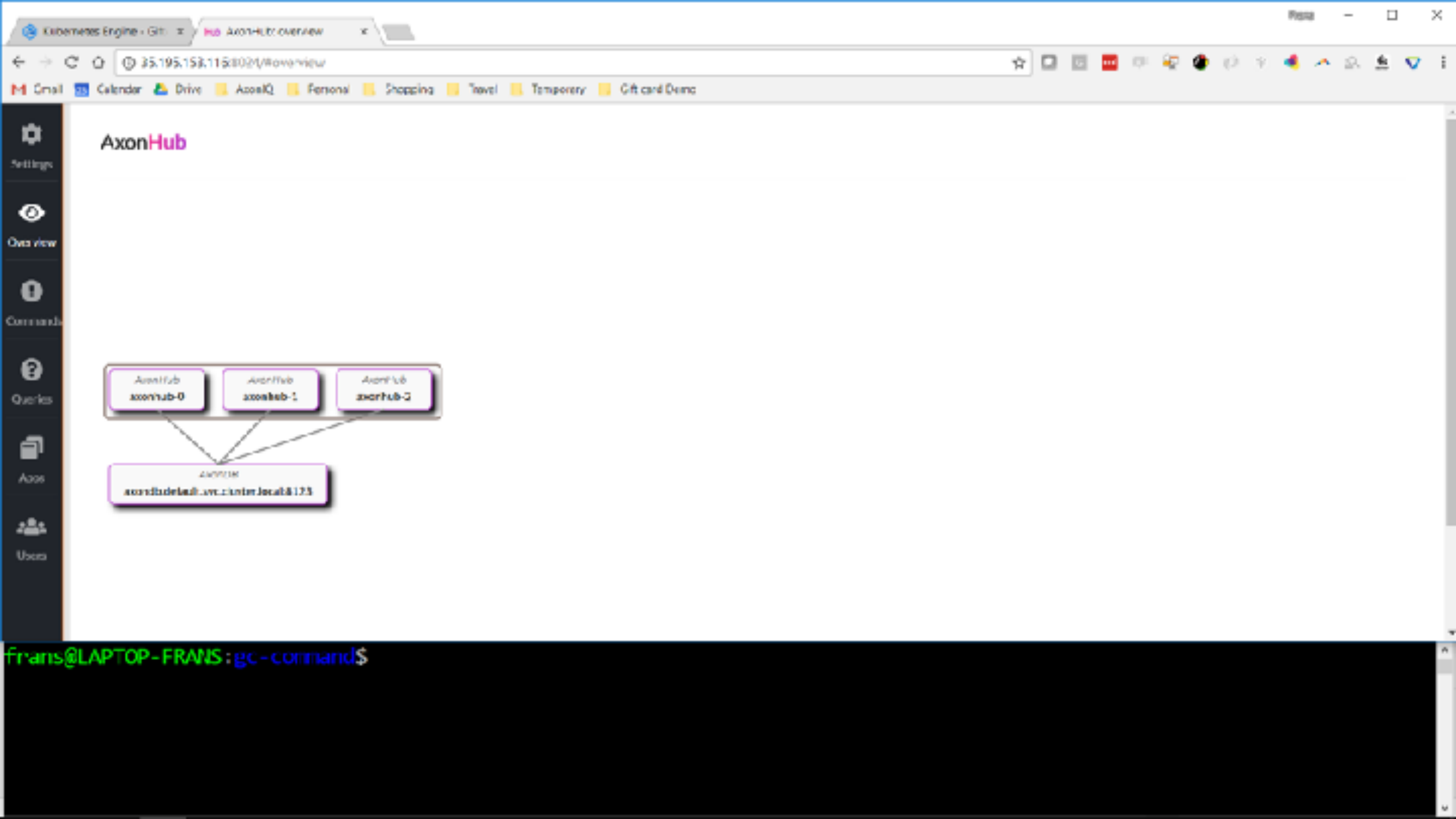
License

Licensee	AxonIQ
Edition	Standard
Expiry date	2018-05-12

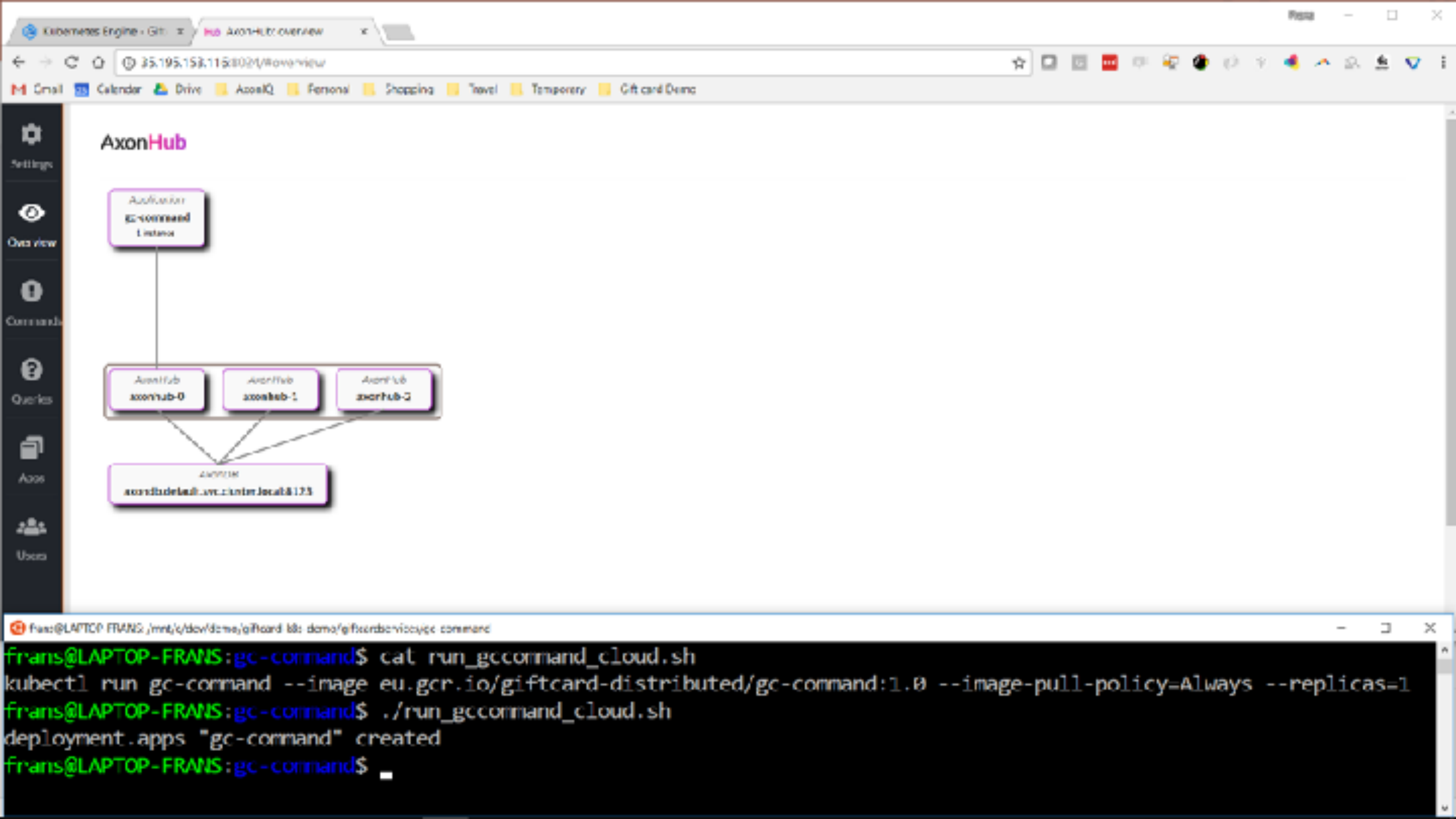
Nodes

Node Name	Host Name	Http Port	GRPC Port
axonhub-0	axonhub-0.axonhub.default.svc.cluster.local	8024	8124

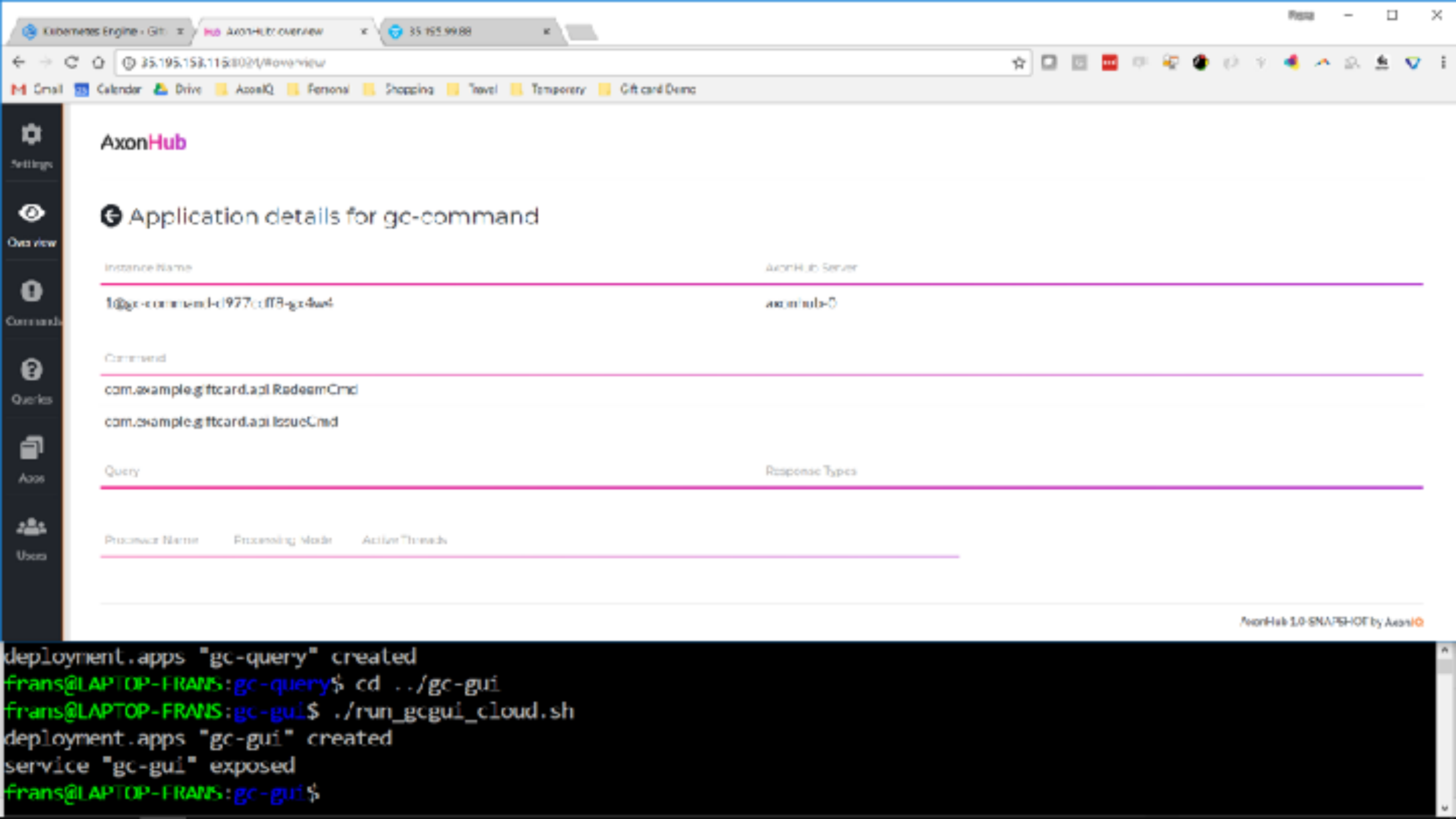
```
frans@LAPTOP-FRANS:gc-command$
```



```
frans@LAPTOP-FRANS:gc-command$
```



```
frans@LAPTOP-FRANS:gc-command$ cat run_gccommand_cloud.sh
kubectl run gc-command --image eu.gcr.io/giftcard-distributed/gc-command:1.0 --image-pull-policy=Always --replicas=1
frans@LAPTOP-FRANS:gc-command$ ./run_gccommand_cloud.sh
deployment.apps "gc-command" created
frans@LAPTOP-FRANS:gc-command$ _
```



AxonHub

Application details for gc-command

Instance Name	AxonHub Server
1@gc-command-d977cdf8-gx4w4	axonhub-0

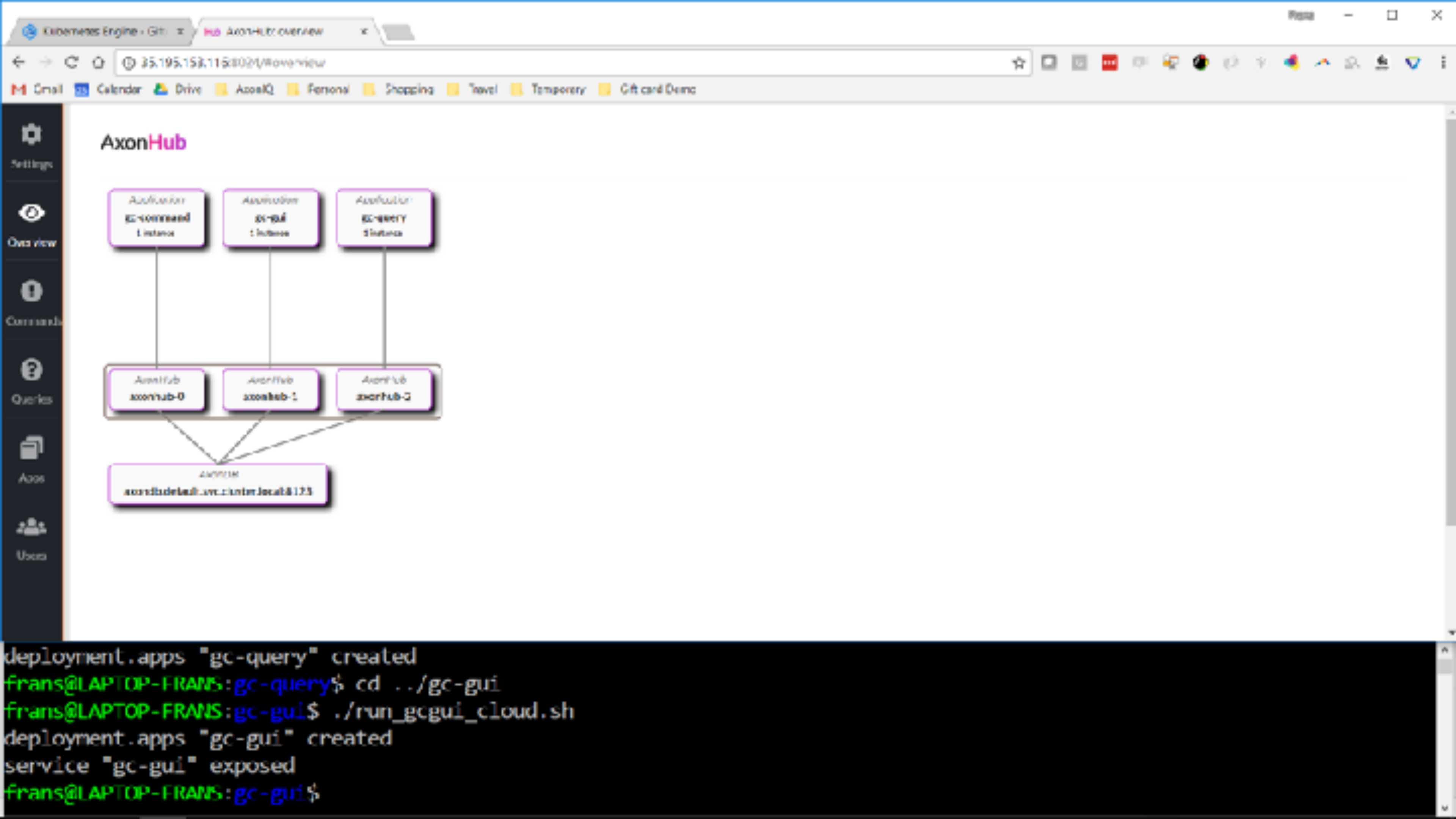
Command
com.example.giftcard.api.RedeemCmd
com.example.giftcard.api.IssueCmd

Query	Response Types

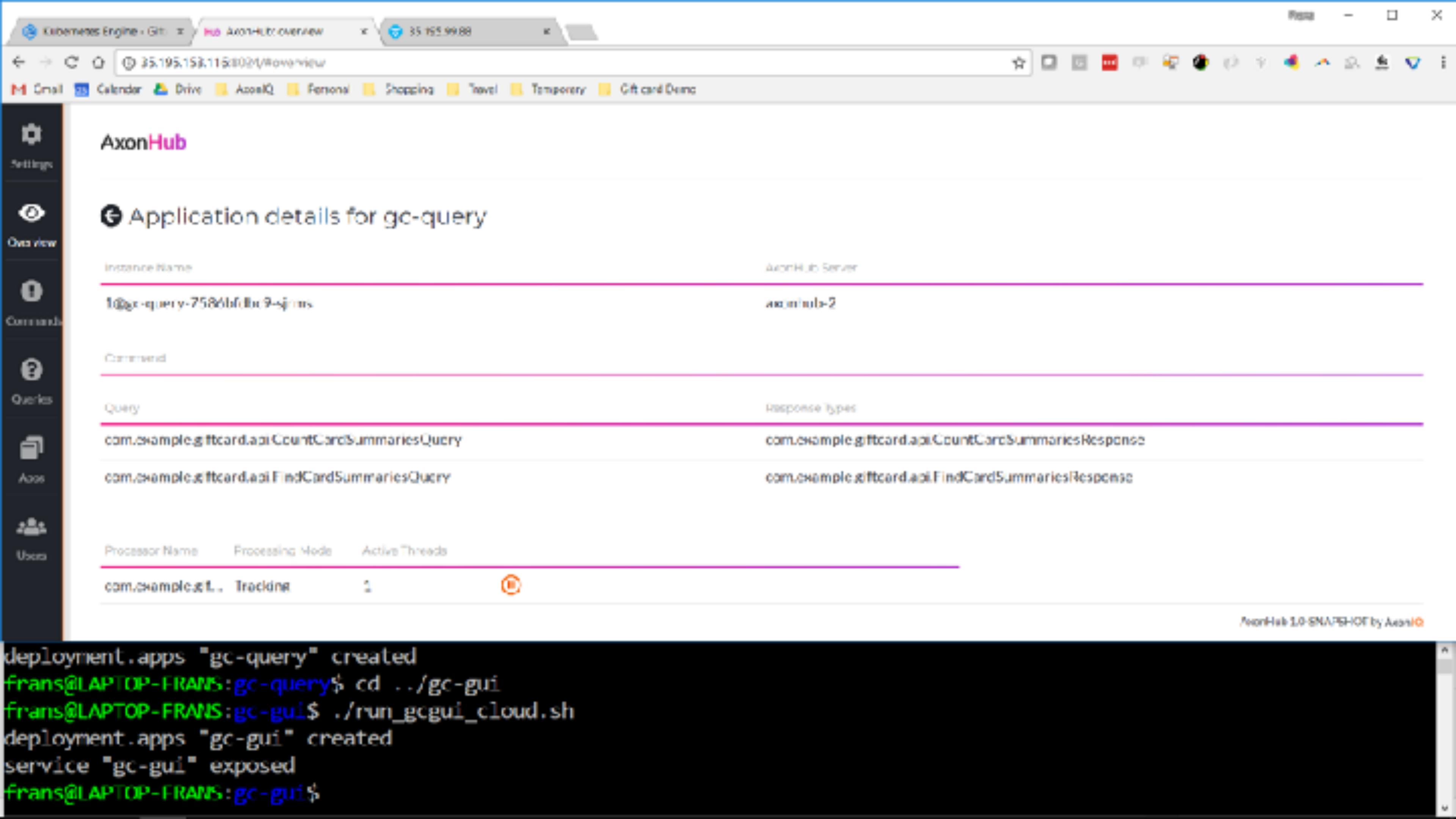
Processor Name	Processing Mode	Active Threads

AxonHub 1.0-SNAPSHOT by AxonIQ

```
deployment.apps "gc-query" created
frans@LAPTOP-FRANS:gc-query$ cd ../gc-gui
frans@LAPTOP-FRANS:gc-gui$ ./run_gcgui_cloud.sh
deployment.apps "gc-gui" created
service "gc-gui" exposed
frans@LAPTOP-FRANS:gc-gui$
```



```
deployment.apps "gc-query" created
frans@LAPTOP-FRANS:gc-query$ cd ../gc-gui
frans@LAPTOP-FRANS:gc-gui$ ./run_gcgui_cloud.sh
deployment.apps "gc-gui" created
service "gc-gui" exposed
frans@LAPTOP-FRANS:gc-gui$
```



Application details for gc-query

Instance Name	AxonHub Server
1@gc-query-7586bf6bc9-sjrms	axonhub-2

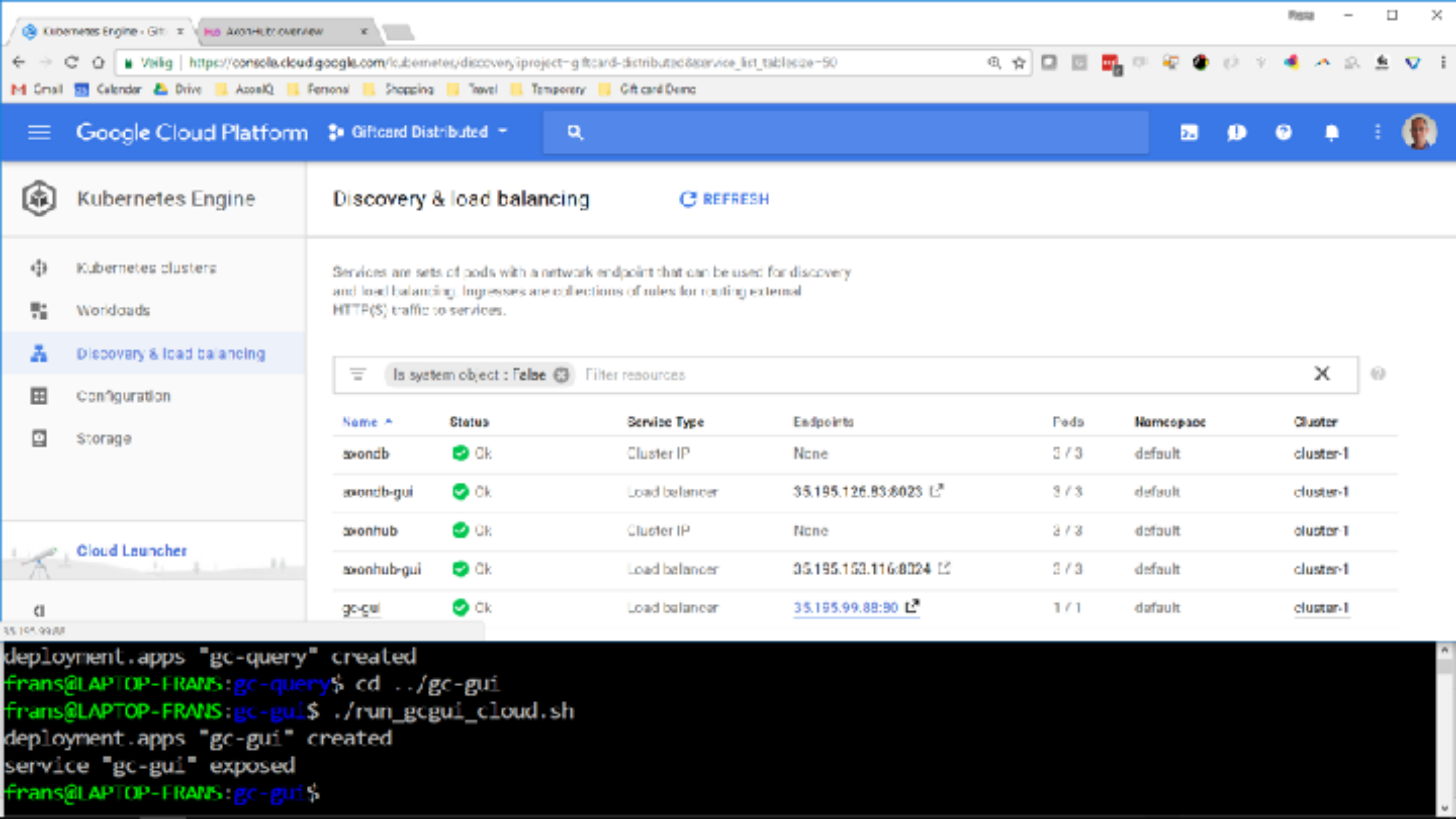
Command

Query	Response Types
com.example.giftcard.api.CountCardSummariesQuery	com.example.giftcard.api.CountCardSummariesResponse
com.example.giftcard.api.FindCardSummariesQuery	com.example.giftcard.api.FindCardSummariesResponse

Processor Name	Processing Mode	Active Threads
com.example.gi...	Tracking	1

AxonHub 1.0-SNAPSHOT by AxonIQ

```
deployment.apps "gc-query" created
frans@LAPTOP-FRANS:gc-query$ cd ../gc-gui
frans@LAPTOP-FRANS:gc-gui$ ./run_gcgui_cloud.sh
deployment.apps "gc-gui" created
service "gc-gui" exposed
frans@LAPTOP-FRANS:gc-gui$
```

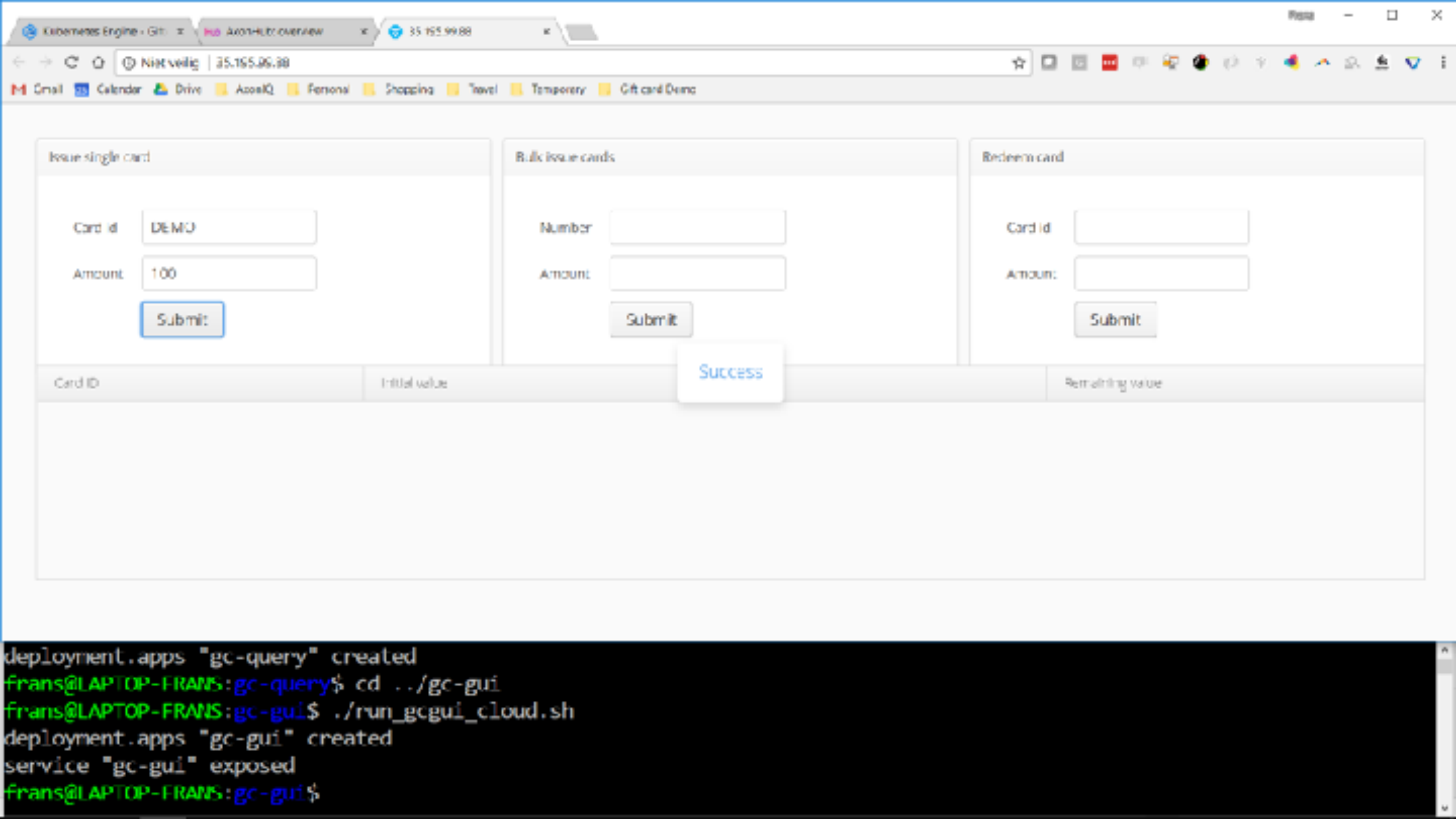


Discovery & load balancing

Services are sets of pods with a network endpoint that can be used for discovery and load balancing. Ingresses are collections of rules for routing external HTTP(S) traffic to services.

Name ▲	Status	Service Type	Endpoints	Pods	Namespace	Cluster
axondb	✓ Ok	Cluster IP	None	3 / 3	default	cluster-1
axondb-gui	✓ Ok	Load balancer	35.195.126.83:8023	3 / 3	default	cluster-1
axonhub	✓ Ok	Cluster IP	None	3 / 3	default	cluster-1
axonhub-gui	✓ Ok	Load balancer	35.195.153.116:8024	3 / 3	default	cluster-1
gc-gui	✓ Ok	Load balancer	35.195.99.88:80	1 / 1	default	cluster-1

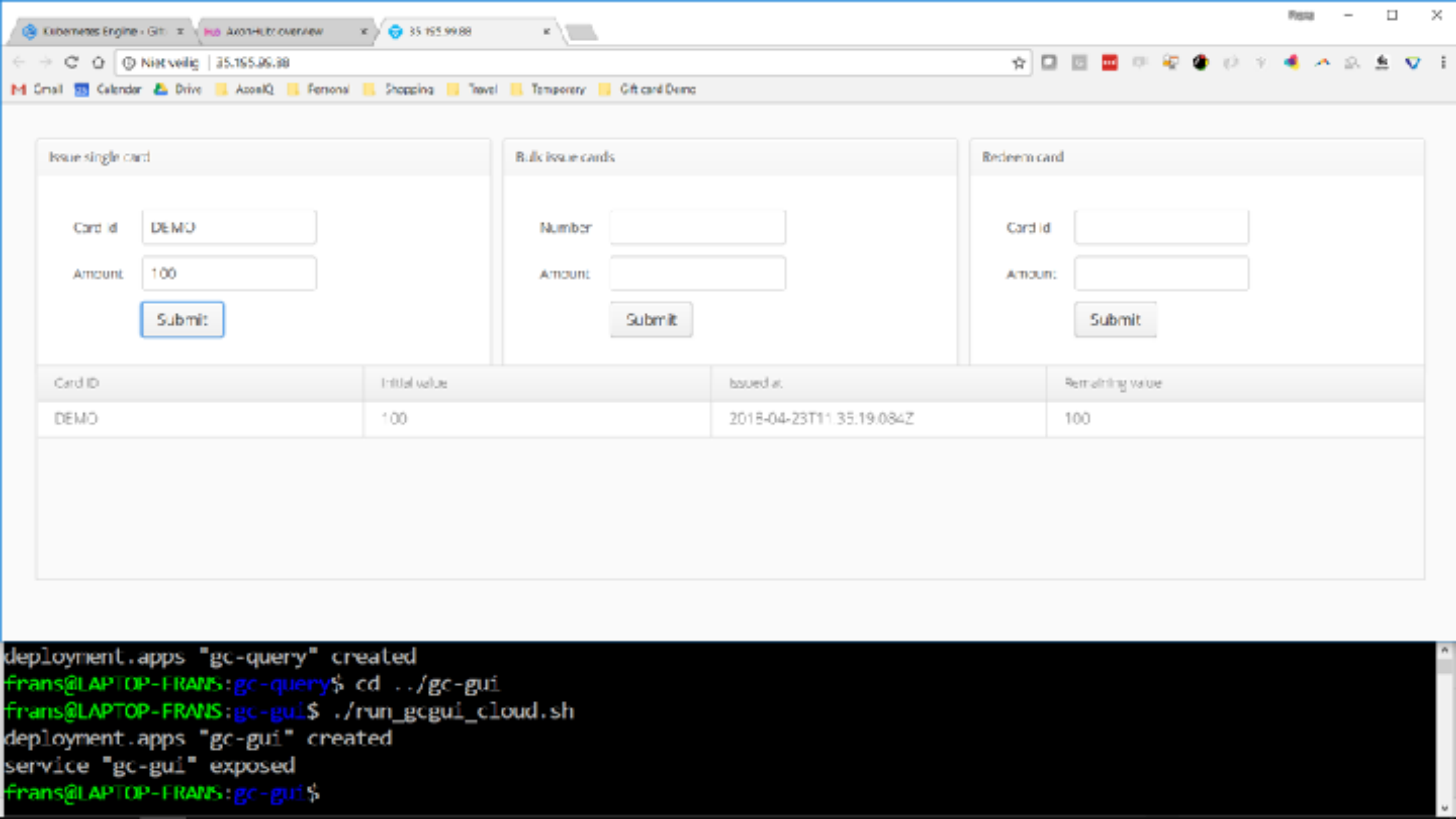
```
deployment.apps "gc-query" created
frans@LAPTOP-FRANS:gc-query$ cd ../gc-gui
frans@LAPTOP-FRANS:gc-gui$ ./run_gcgui_cloud.sh
deployment.apps "gc-gui" created
service "gc-gui" exposed
frans@LAPTOP-FRANS:gc-gui$
```



Card id: DEMO

Amount: 100

```
deployment.apps "gc-query" created
frans@LAPTOP-FRANS:gc-query$ cd ../gc-gui
frans@LAPTOP-FRANS:gc-gui$ ./run_gcgui_cloud.sh
deployment.apps "gc-gui" created
service "gc-gui" exposed
frans@LAPTOP-FRANS:gc-gui$
```



Issue single card

Card id: DEMO

Amount: 100

Submit

Bulk issue cards

Number:

Amount:

Submit

Redeem card

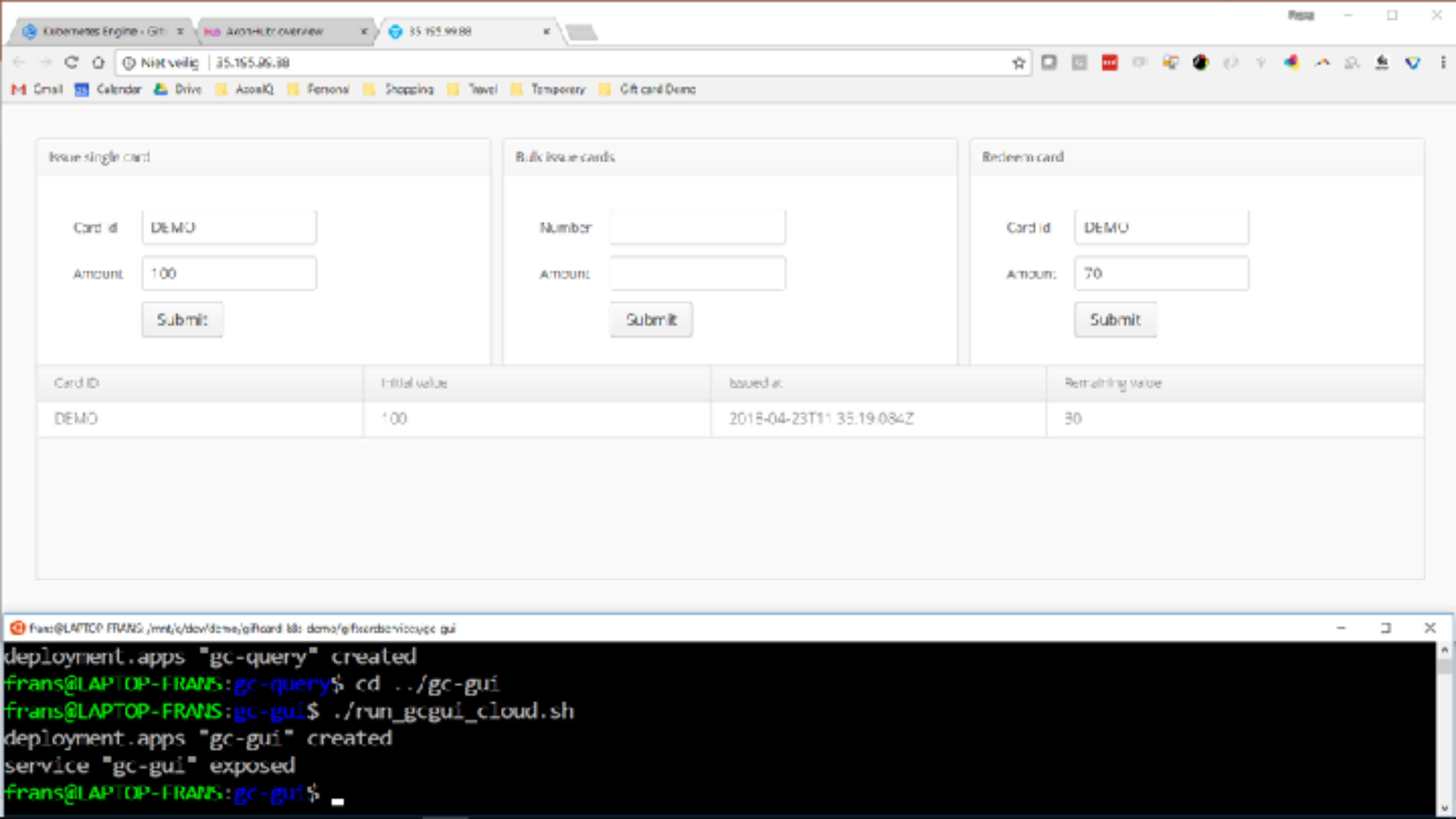
Card id:

Amount:

Submit

Card ID	Initial value	Issued at	Remaining value
DEMO	100	2018-04-23T11:35:19.084Z	100

```
deployment.apps "gc-query" created
frans@LAPTOP-FRANS:gc-query$ cd ../gc-gui
frans@LAPTOP-FRANS:gc-gui$ ./run_gcgui_cloud.sh
deployment.apps "gc-gui" created
service "gc-gui" exposed
frans@LAPTOP-FRANS:gc-gui$
```



Issue single card

Card id: DEMO

Amount: 100

Submit

Bulk issue cards

Number:

Amount:

Submit

Redeem card

Card id: DEMO

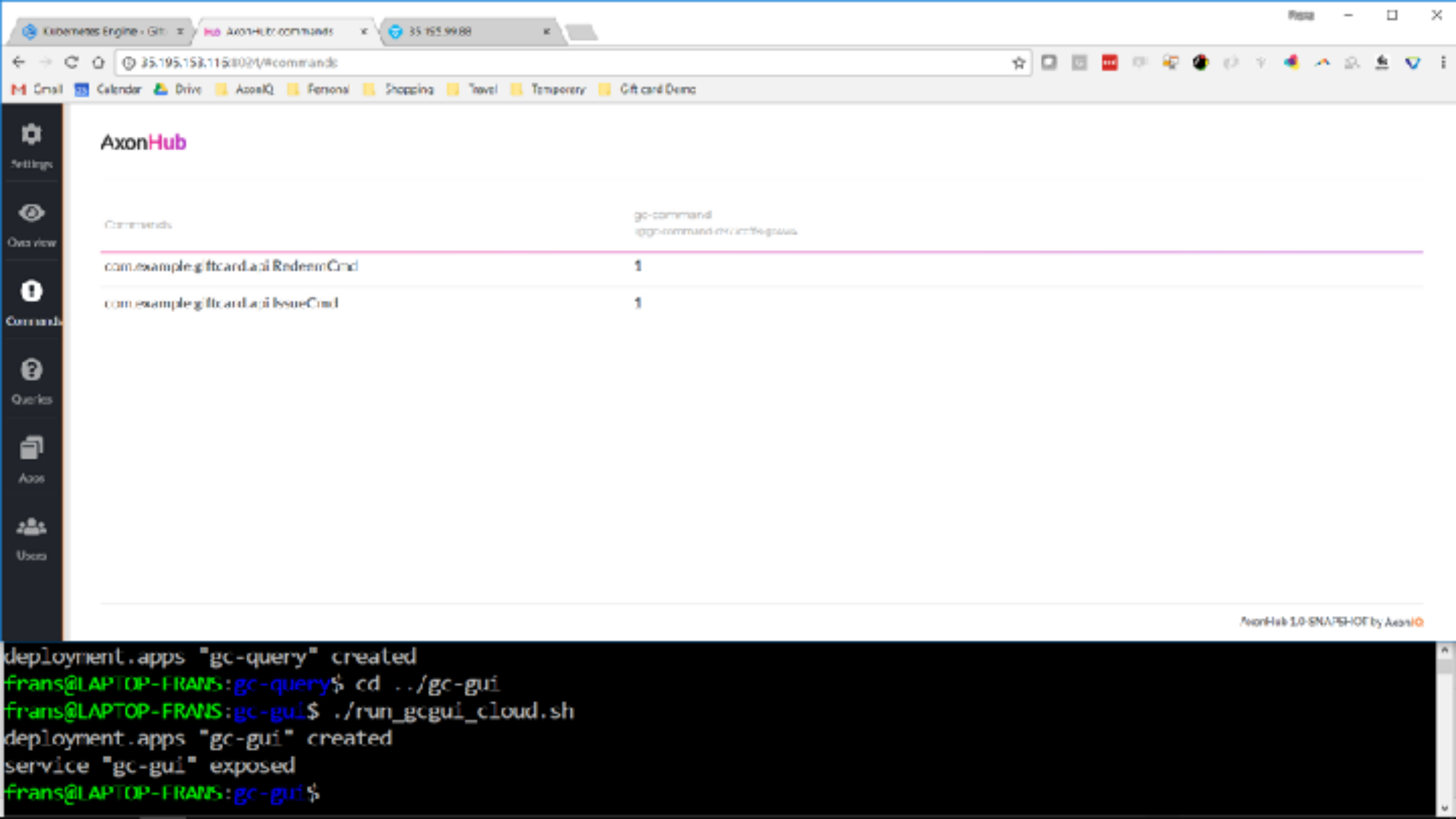
Amount: 70

Submit

Card ID	Initial value	Issued at	Remaining value
DEMO	100	2018-04-23T11:35:19.084Z	30

frans@LAPTOP-FRANS: /mnt/c/dev/demo/giftcard-k8s-demo/giftcardservice/gc-gui

```
deployment.apps "gc-query" created
frans@LAPTOP-FRANS:gc-query$ cd ../gc-gui
frans@LAPTOP-FRANS:gc-gui$ ./run_gcgui_cloud.sh
deployment.apps "gc-gui" created
service "gc-gui" exposed
frans@LAPTOP-FRANS:gc-gui$ _
```



AxonHub

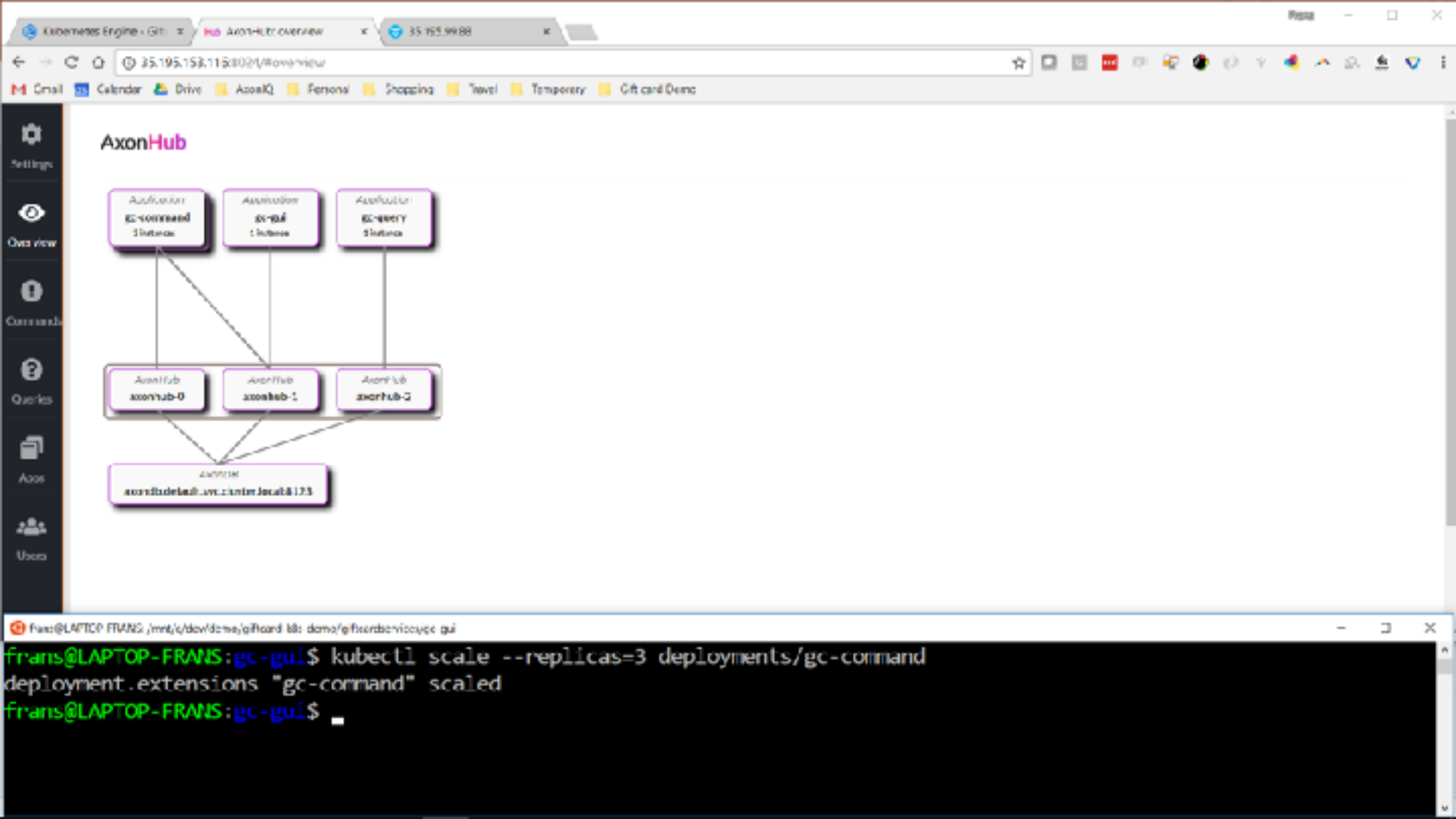
Commands	gc-command [illegible]
com.example.giftcard.api.RedeemCmd	1
com.example.giftcard.api.IssueCmd	1

AxonHub 1.0-SNAPSHOT by AxonIQ

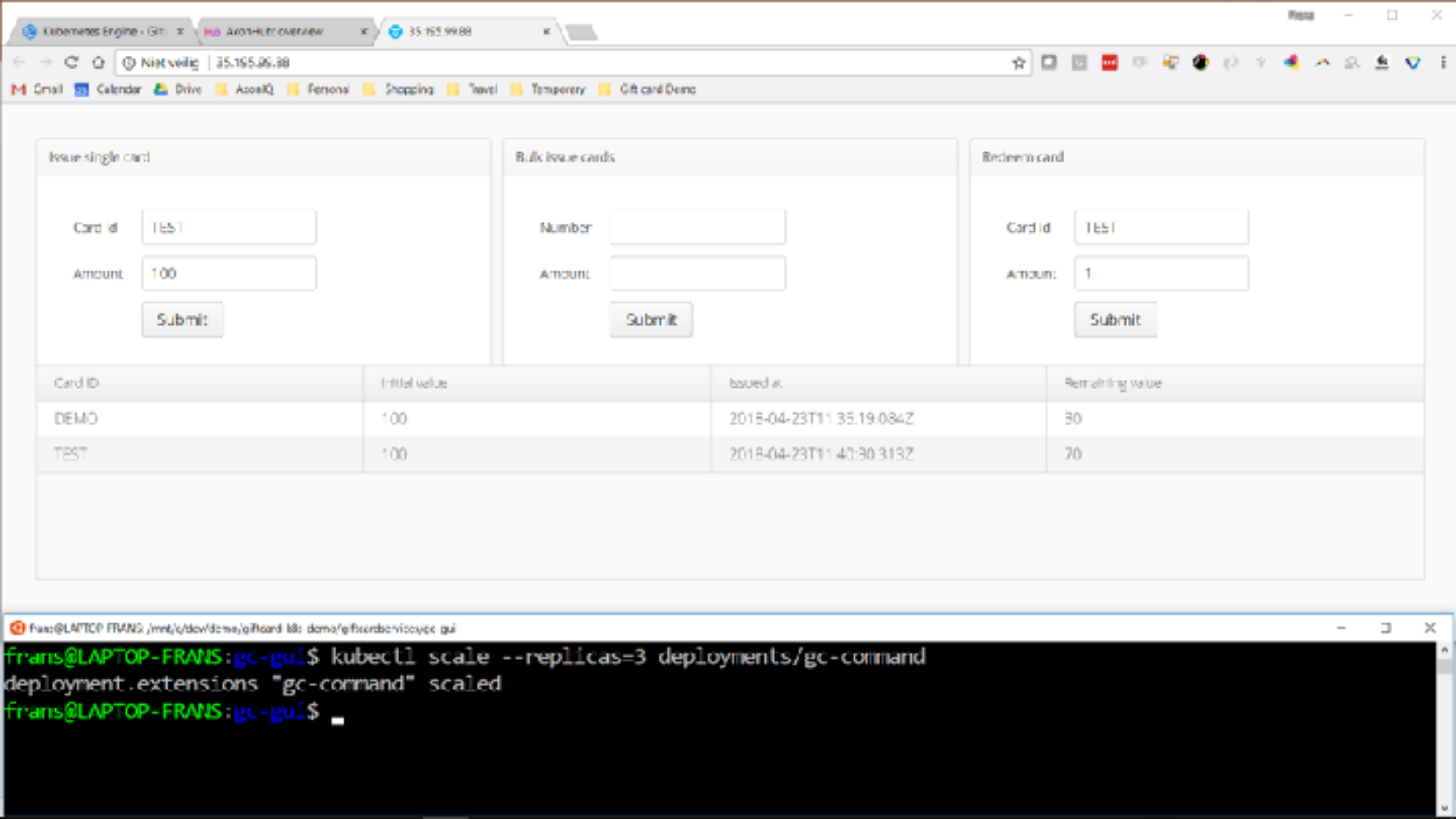
```
deployment.apps "gc-query" created
frans@LAPTOP-FRANS:gc-query$ cd ../gc-gui
frans@LAPTOP-FRANS:gc-gui$ ./run_gcgui_cloud.sh
deployment.apps "gc-gui" created
service "gc-gui" exposed
frans@LAPTOP-FRANS:gc-gui$
```

Demo

Scaling out



```
frans@LAPTOP-FRANS:gc-gui$ kubectl scale --replicas=3 deployments/gc-command
deployment.extensions "gc-command" scaled
frans@LAPTOP-FRANS:gc-gui$ _
```



Issue single card

Card id: TEST

Amount: 100

Submit

Bulk issue cards

Number:

Amount:

Submit

Redeem card

Card id: TEST

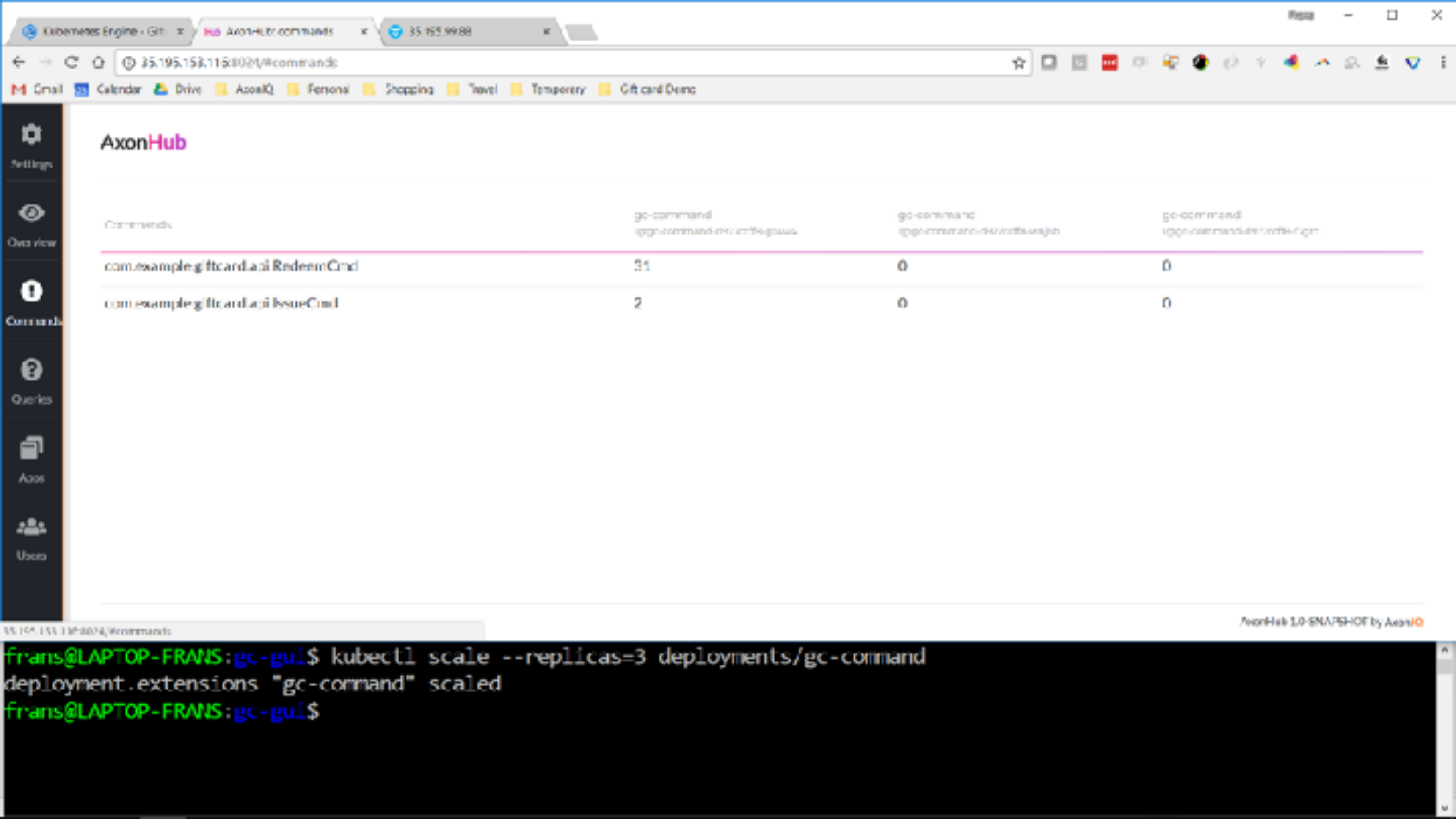
Amount: 1

Submit

Card ID	Initial value	Issued at	Remaining value
DEMO	100	2018-04-23T11:35:19.084Z	30
TEST	100	2018-04-23T11:40:30.313Z	70

frans@LAPTOP-FRANS: /mnt/c/dev/demo/giftcard-kb-demo/giftcardservice/gc-gui

```
frans@LAPTOP-FRANS:gc-gui$ kubectl scale --replicas=3 deployments/gc-command
deployment.extensions "gc-command" scaled
frans@LAPTOP-FRANS:gc-gui$ _
```

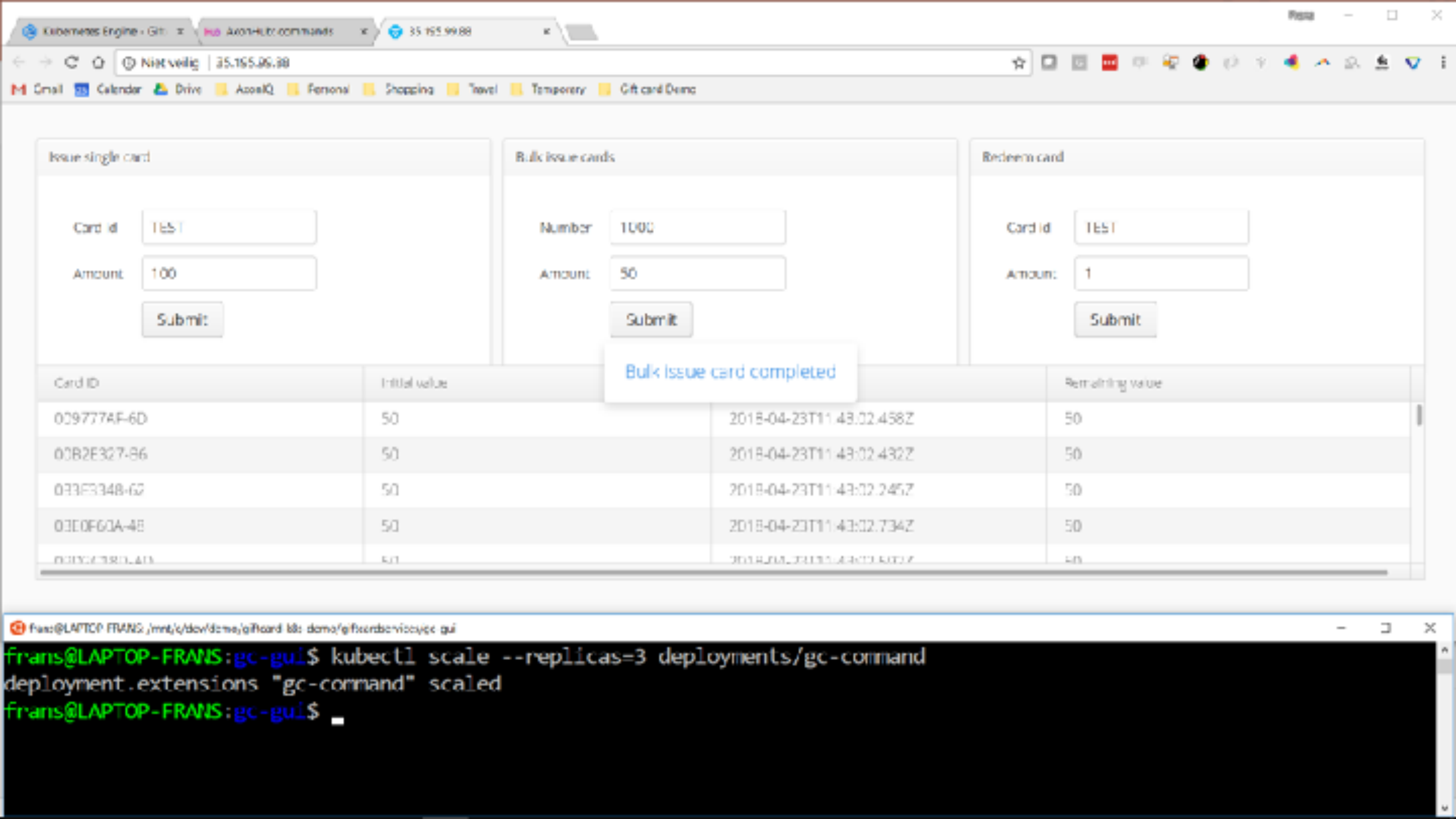


AxonHub

Commands	gc-command [illegible]	gc-command [illegible]	gc-command [illegible]
com.example.giftcard.api.RedeemCmd	31	0	0
com.example.giftcard.api.IssueCmd	2	0	0

AxonHub 1.0-SNAPSHOT by AxonIQ

```
frans@LAPTOP-FRANS:gc-gui$ kubectl scale --replicas=3 deployments/gc-command
deployment.extensions "gc-command" scaled
frans@LAPTOP-FRANS:gc-gui$
```



Issue single card

Card id: TEST

Amount: 100

Submit

Bulk issue cards

Number: 1000

Amount: 50

Submit

Redeem card

Card id: TEST

Amount: 1

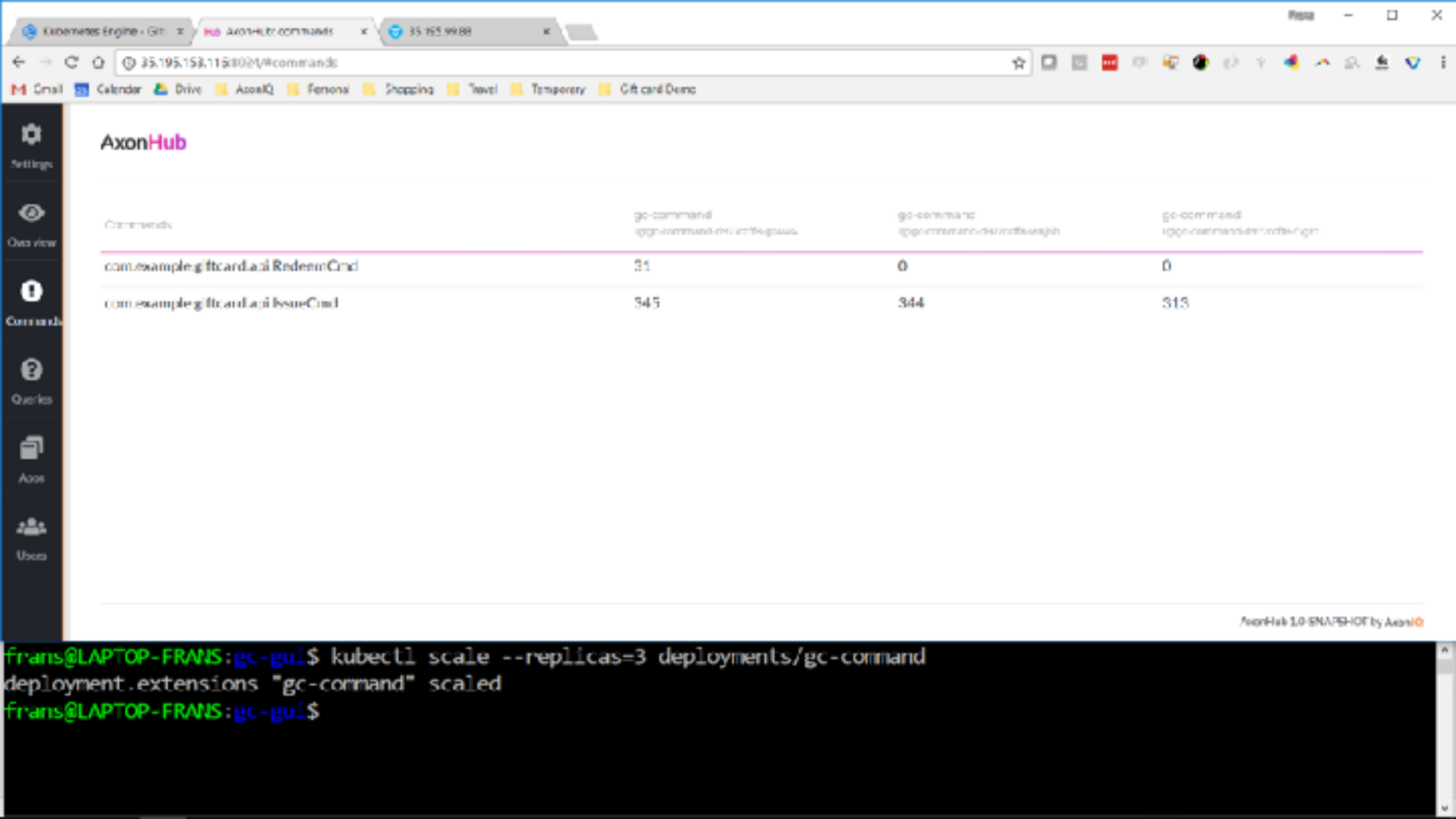
Submit

Bulk issue card completed

Card ID	Initial value		Remaining value
009777AF-6D	50	2018-04-23T11:43:02.458Z	50
00B2E327-86	50	2018-04-23T11:43:02.432Z	50
033E3348-62	50	2018-04-23T11:43:02.245Z	50
03E0F60A-48	50	2018-04-23T11:43:02.734Z	50
[illegible]	50	[illegible]	50

frans@LAPTOP-FRANS: /mnt/c/dev/demo/giftcard-k8s-demo/giftcardservices/gc-gui

```
frans@LAPTOP-FRANS:gc-gui$ kubectl scale --replicas=3 deployments/gc-command
deployment.extensions "gc-command" scaled
frans@LAPTOP-FRANS:gc-gui$ _
```



AxonHub

Commands	gc-command [illegible]	gc-command [illegible]	gc-command [illegible]
com.example.giftcard.api.RedeemCmd	31	0	0
com.example.giftcard.api.IssueCmd	345	344	313

AxonHub 1.0-SNAPSHOT by AxonIQ

```
frans@LAPTOP-FRANS:gc-gui$ kubectl scale --replicas=3 deployments/gc-command
deployment.extensions "gc-command" scaled
frans@LAPTOP-FRANS:gc-gui$
```

Please

Remember to rate this session

Thank you!