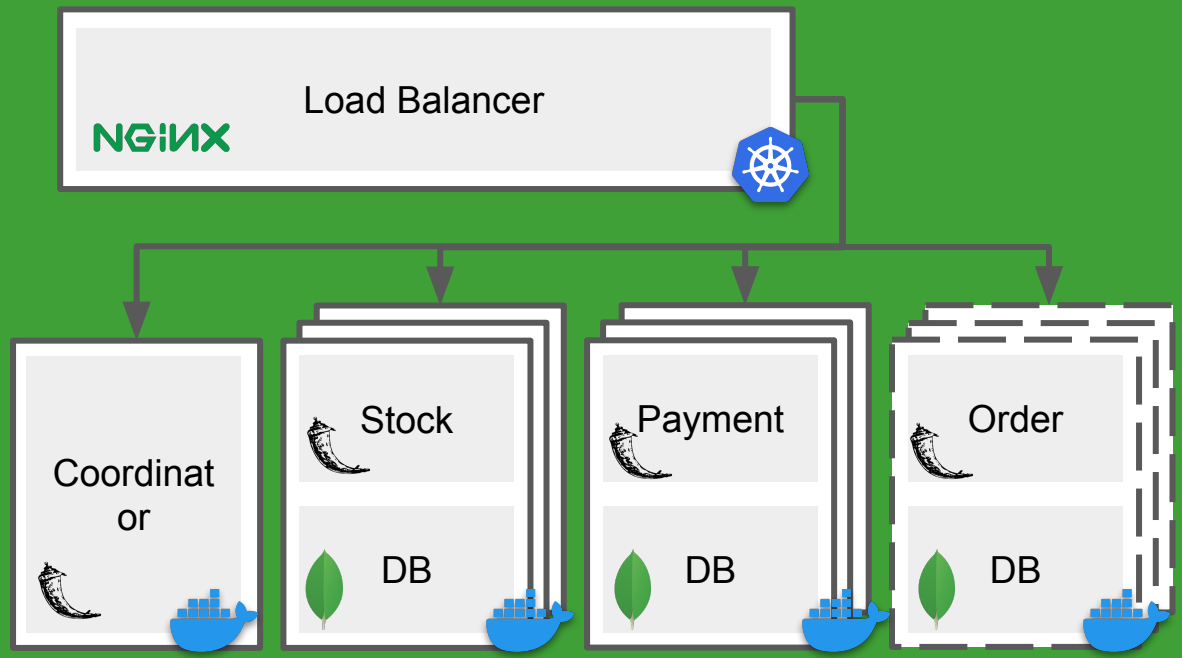
Group 9

Web scale Data Management
Final Presentation

Overview of the Architecture



mongoDB.

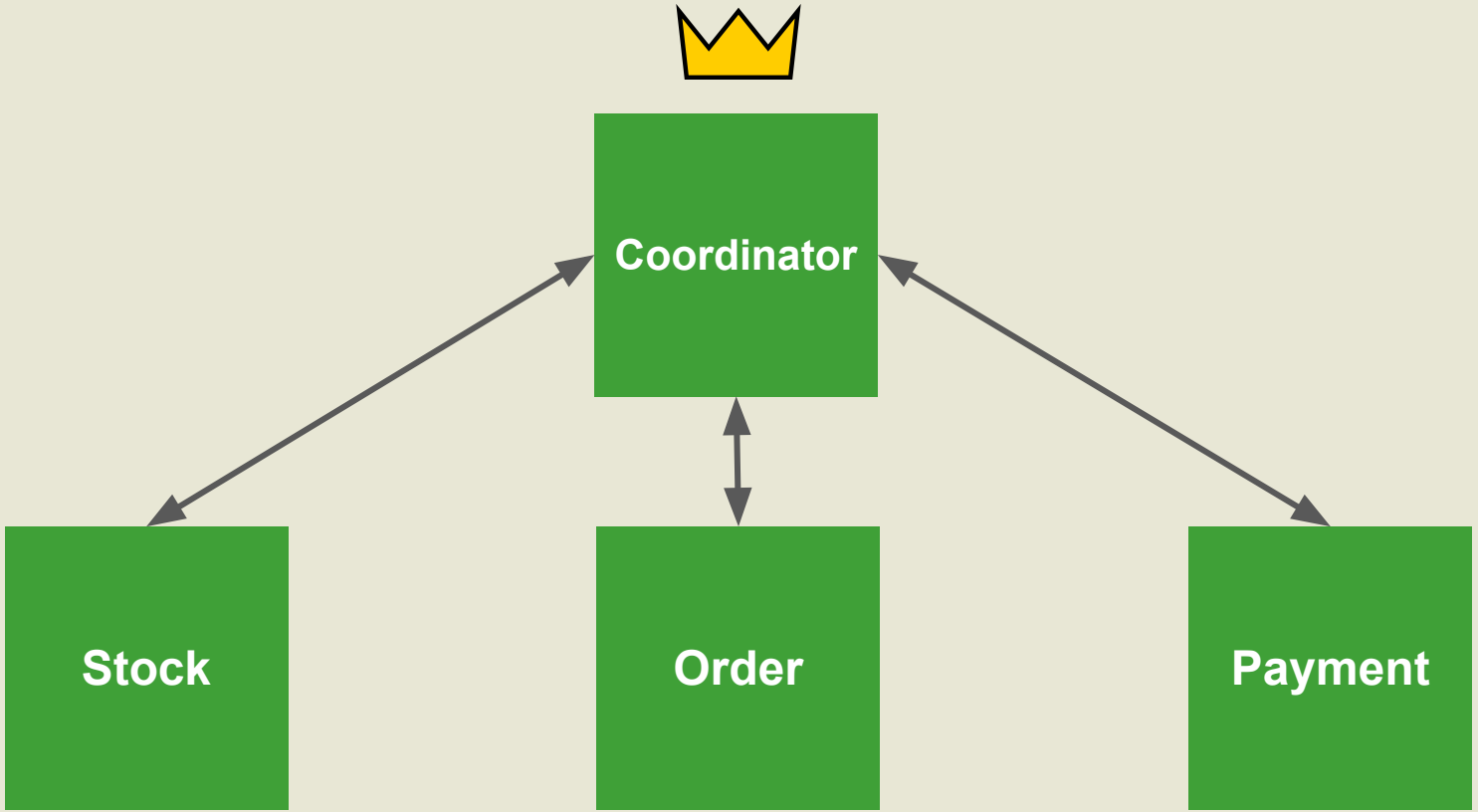
Flask

NGINX

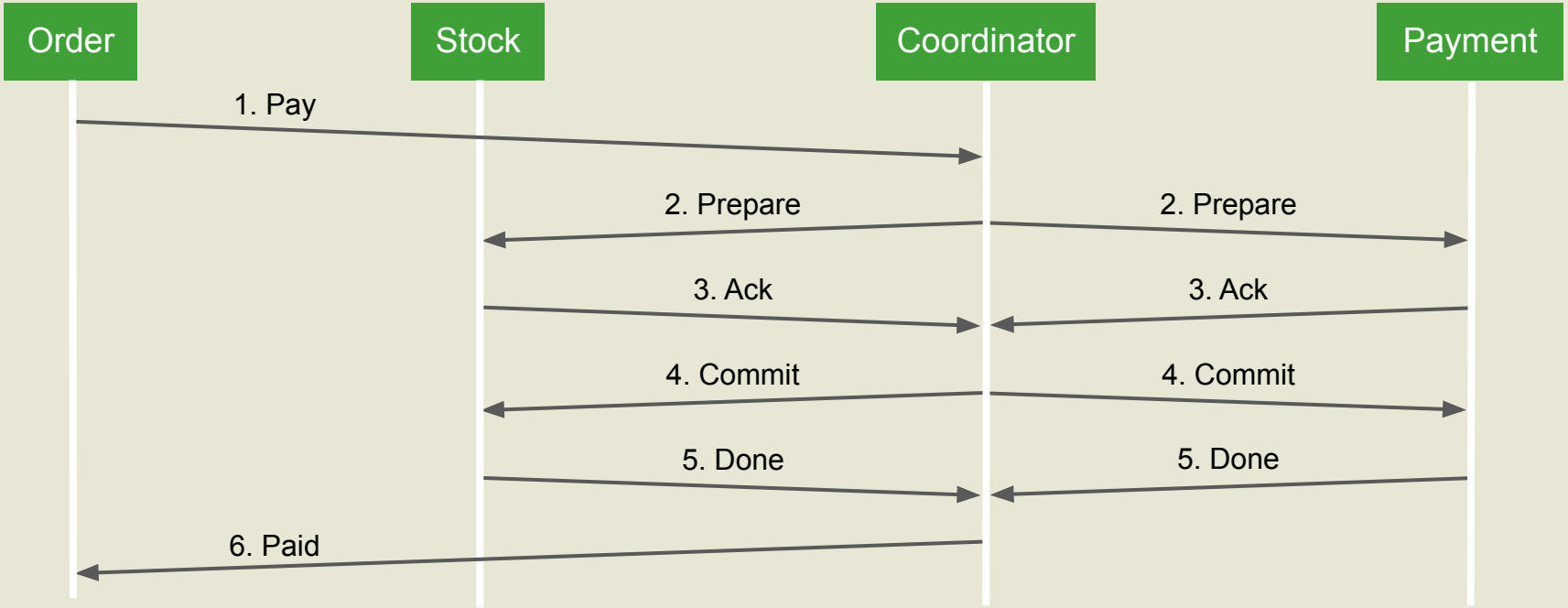
Distributed transactions

How do we guarantee consistency
between the stock and payment service?

2 Phase Commit


Coordinator
Stock
Order
Payment

2PC


Order
Stock
Coordinator
Payment
1. Pay
2. Prepare
2. Prepare
3. Ack
3. Ack
4. Commit
4. Commit
5. Done
5. Done
6. Paid

Problem

2PC is a blocking protocol

Items and users are locked during payment

Bad Availability

Solution

Modify the 2PC protocol to not block the items

Comparison modified 2PC

Standard 2PC

Consistency

```
verify - Stock service inconsistencies in the logs: -64
verify - Stock service inconsistencies in the database: -64
verify - Payment service inconsistencies in the logs: 64
verify - Payment service inconsistencies in the database: 64.0
```

Modified 2PC

Consistency

```
verify - Stock service inconsistencies in the logs: 0
verify - Stock service inconsistencies in the database: 0
verify - Payment service inconsistencies in the logs: 0
verify - Payment service inconsistencies in the database: 0.0
```

Fault tolerance

Stock	Payment	Order
Replicas	Replicas	Shards
Paxos	Read Quorum Write All	User id

Durability
Consistency

Availability

Consistency strategies

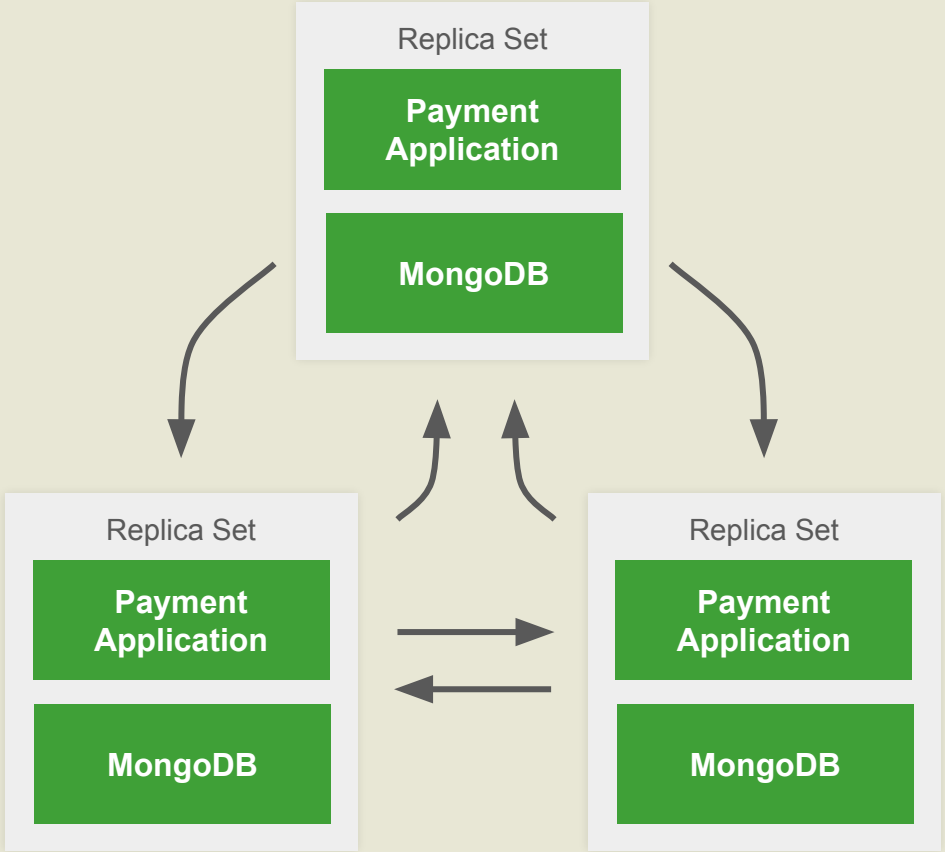
Why we use **PAXOS** and not **2PC** for Payment service?

Why we use **eventual consistency** for Order service?

Why we use consistency via **quorums** for stock service?

How we use PAXOS?

- Payment Service
  - Durability
  - Availability
  - Consistency



Deployment

- Set amount of pods per service
- Magic super-script.sh

minikube