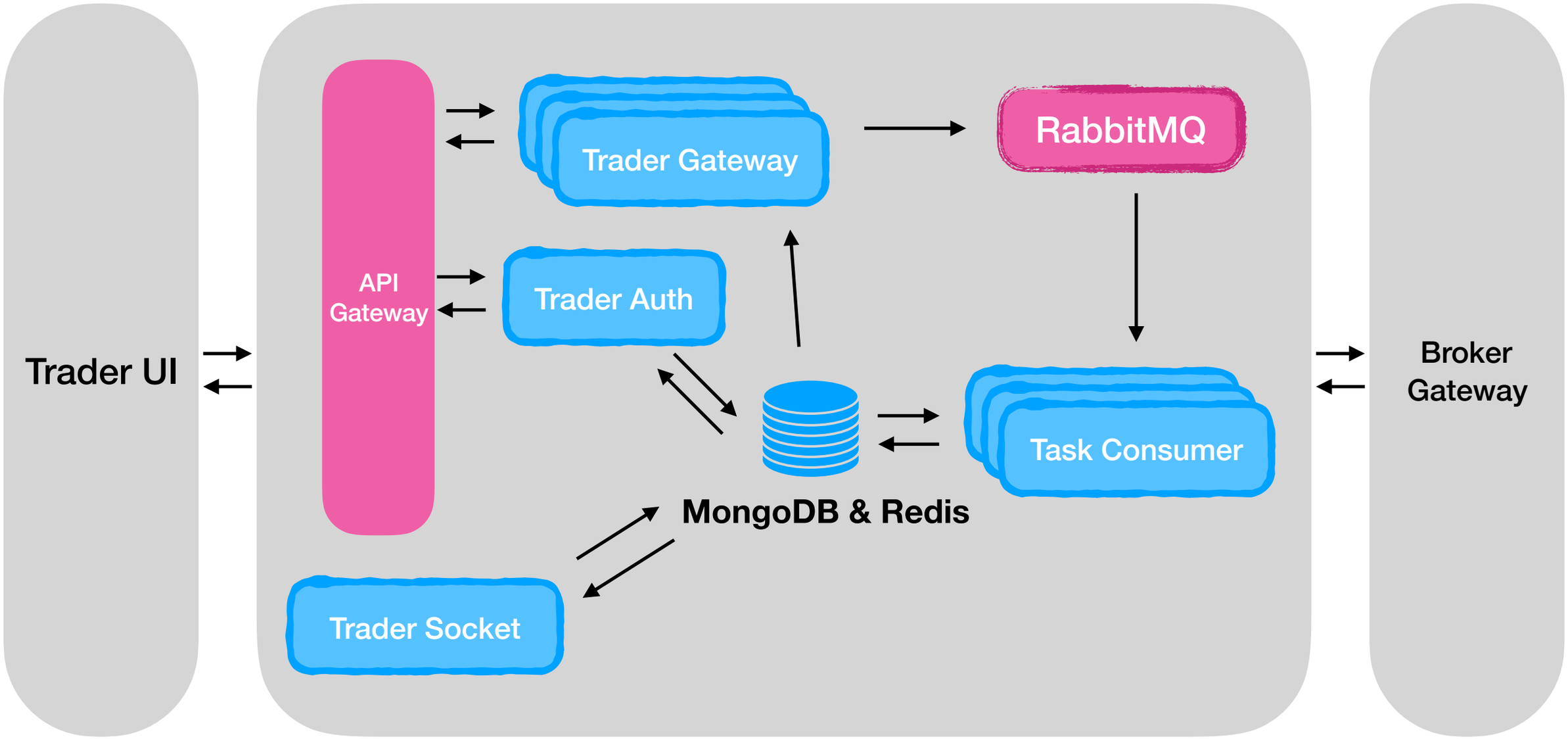
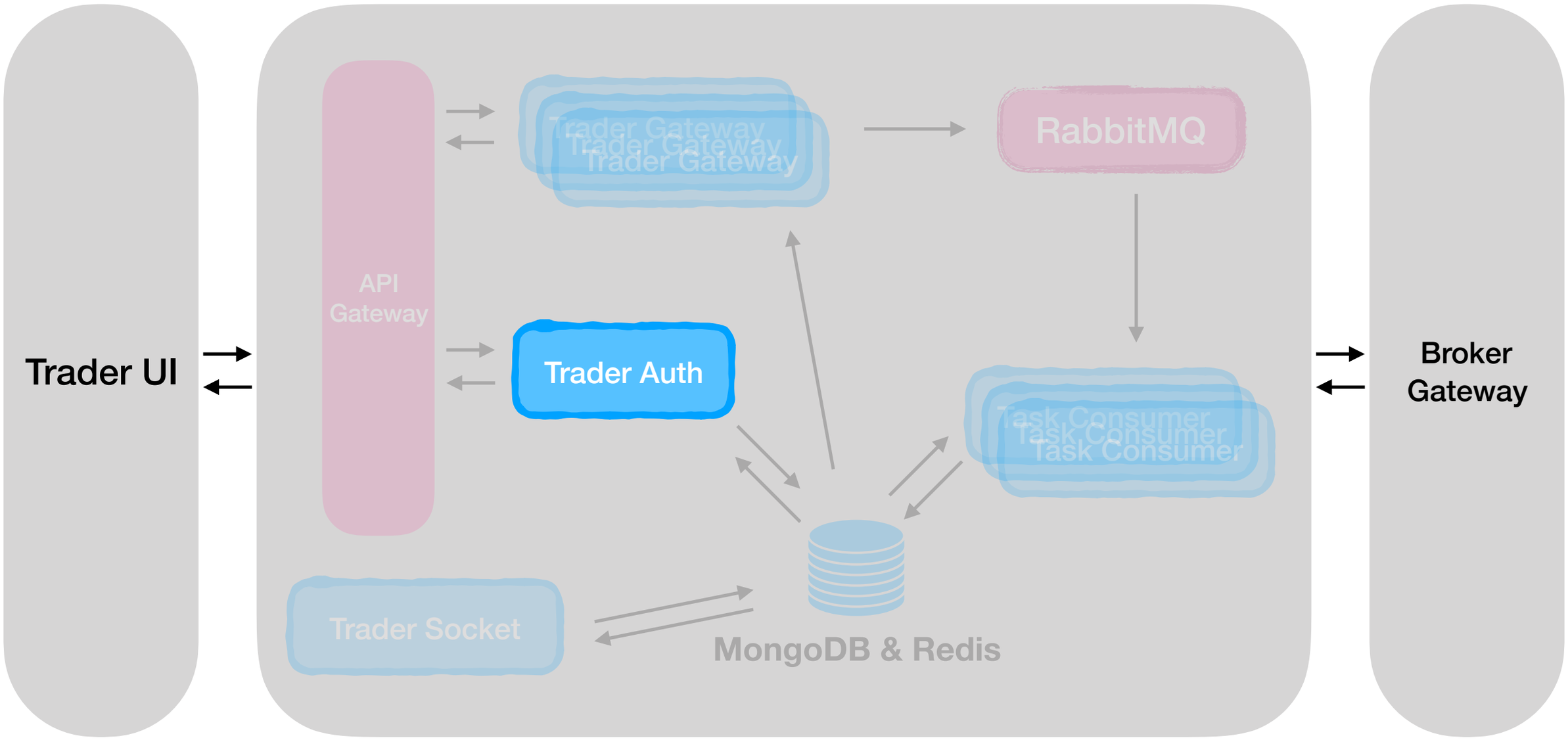
Trader Component



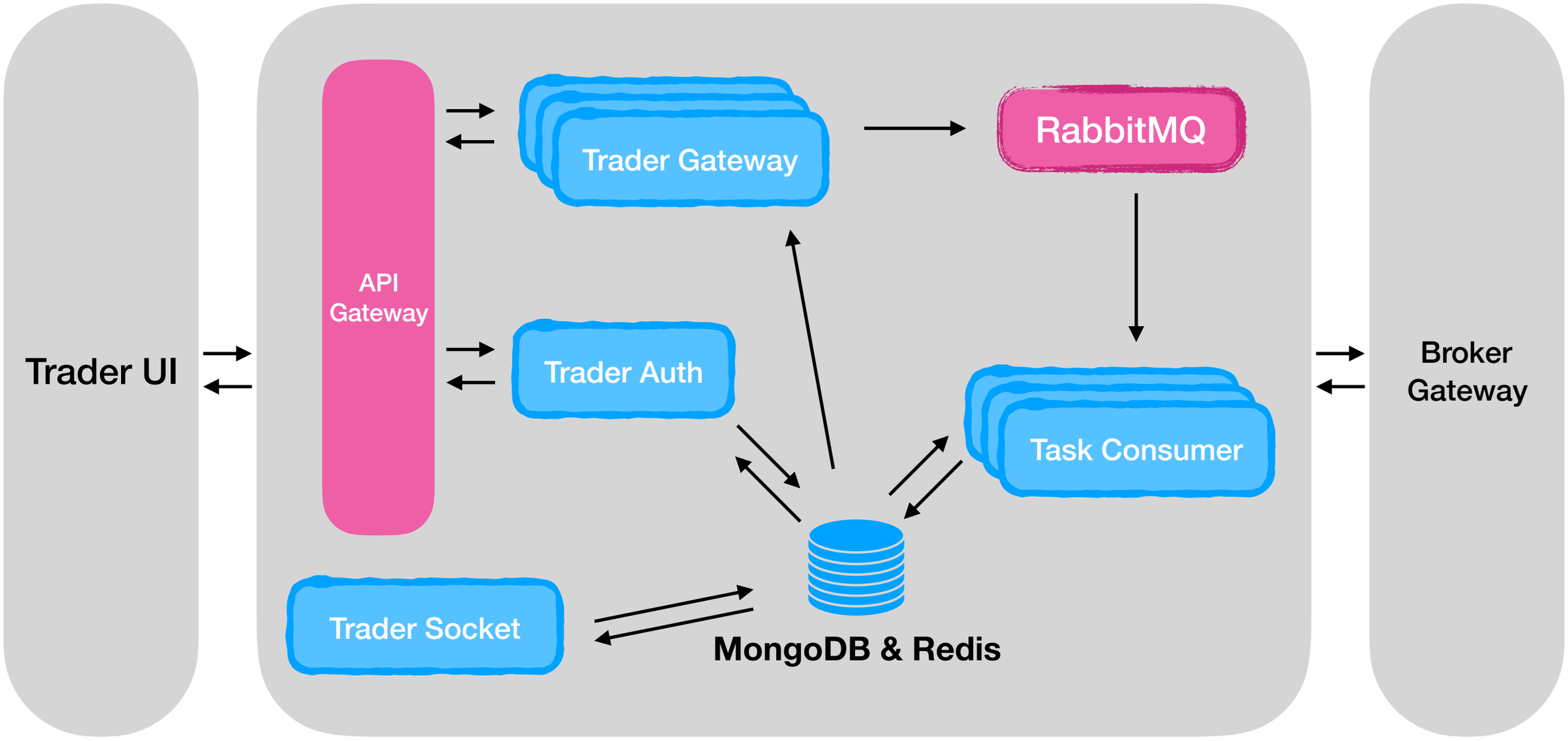
Trader Component



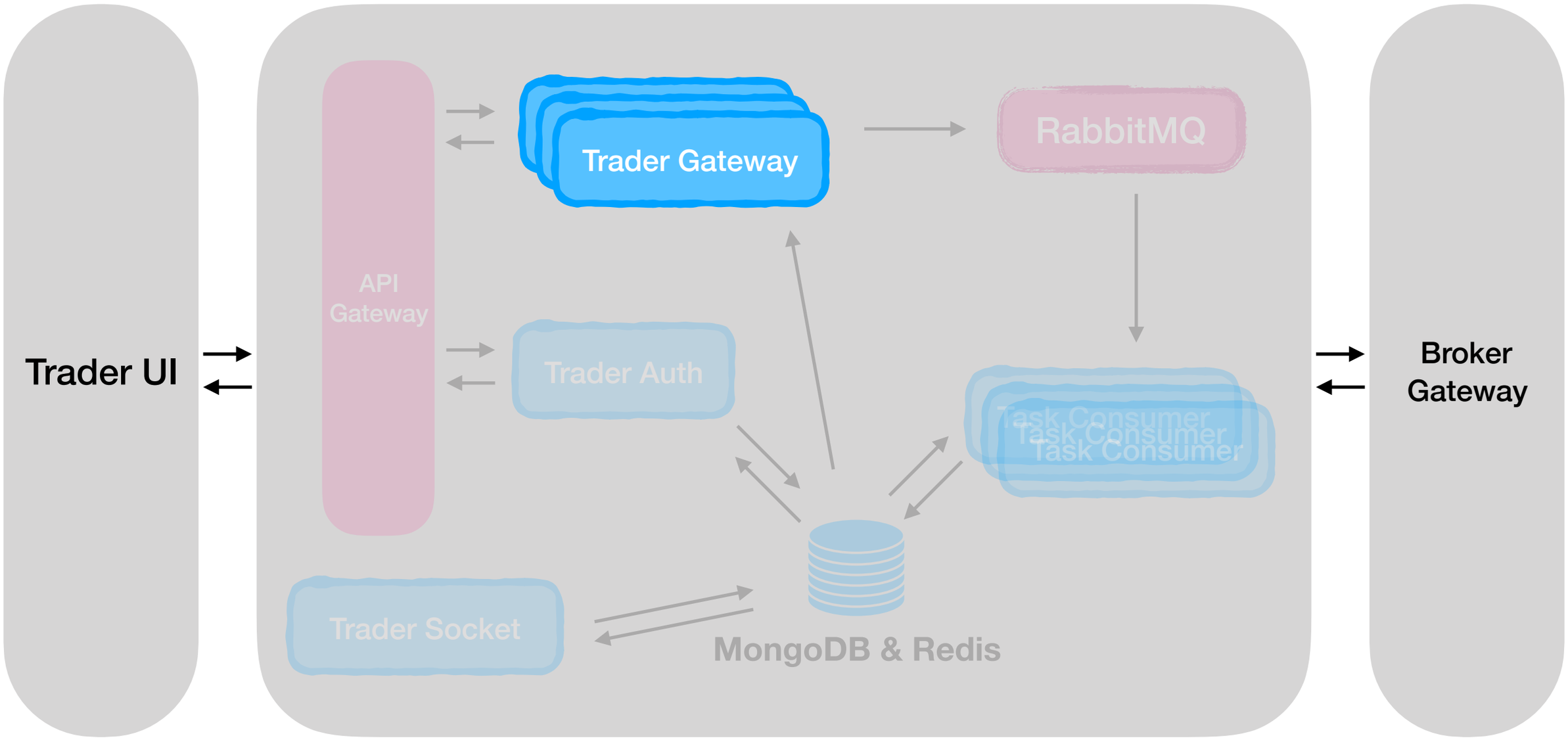
Trader Auth

- Login - JWT
- User infomation

Trader Component



Trader Component



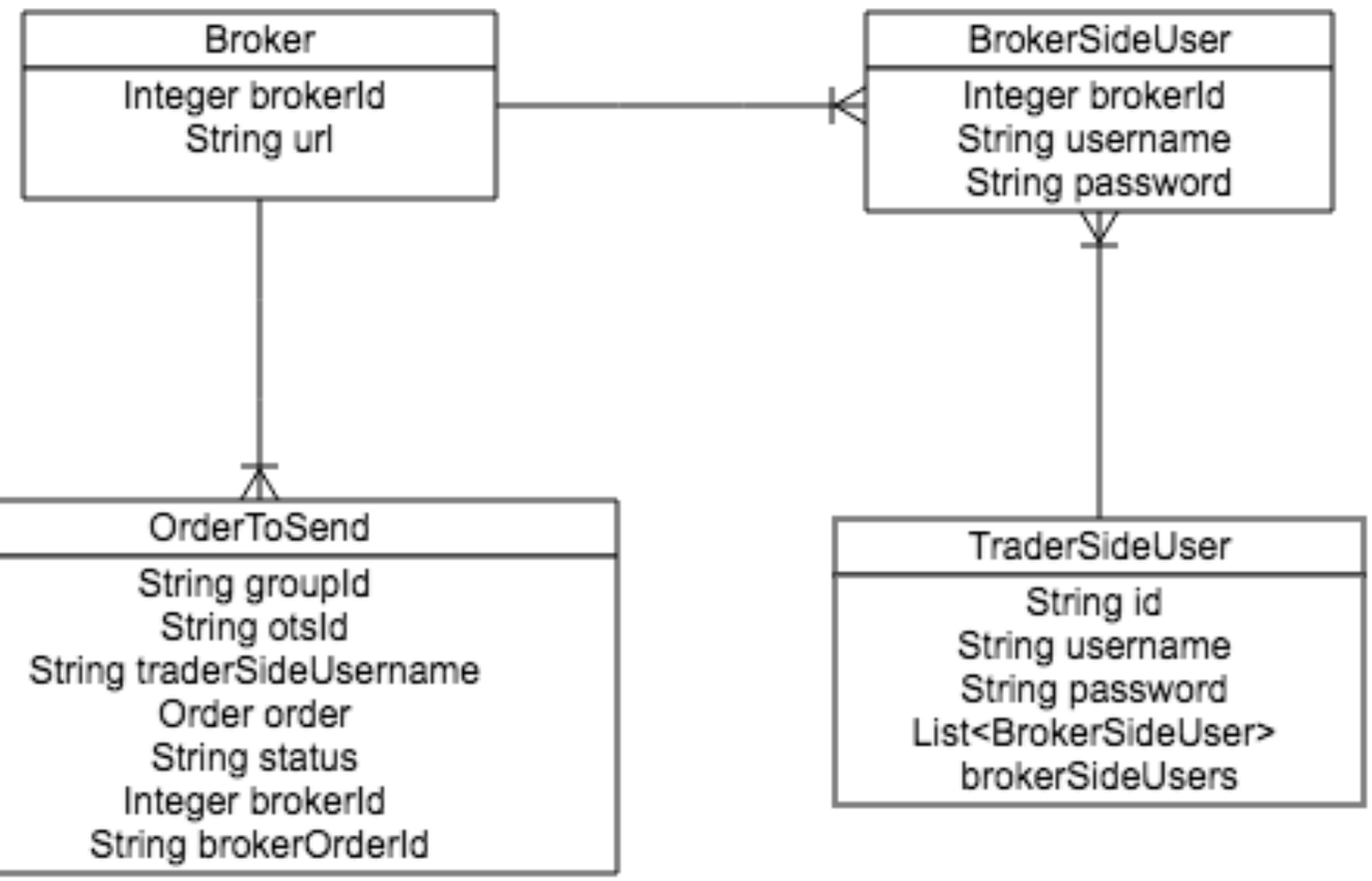
Trader Gateway

- Multiple Brokers

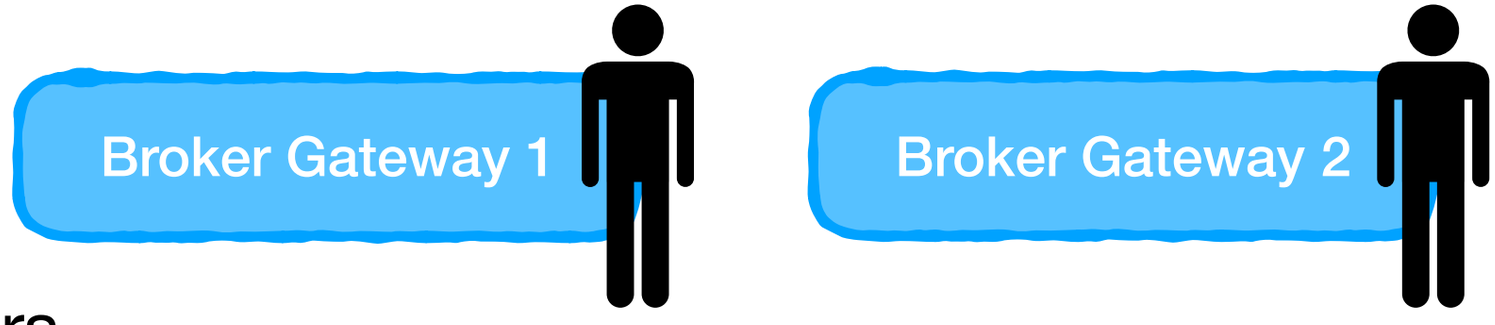
Trader Gateway

- Multiple Brokers

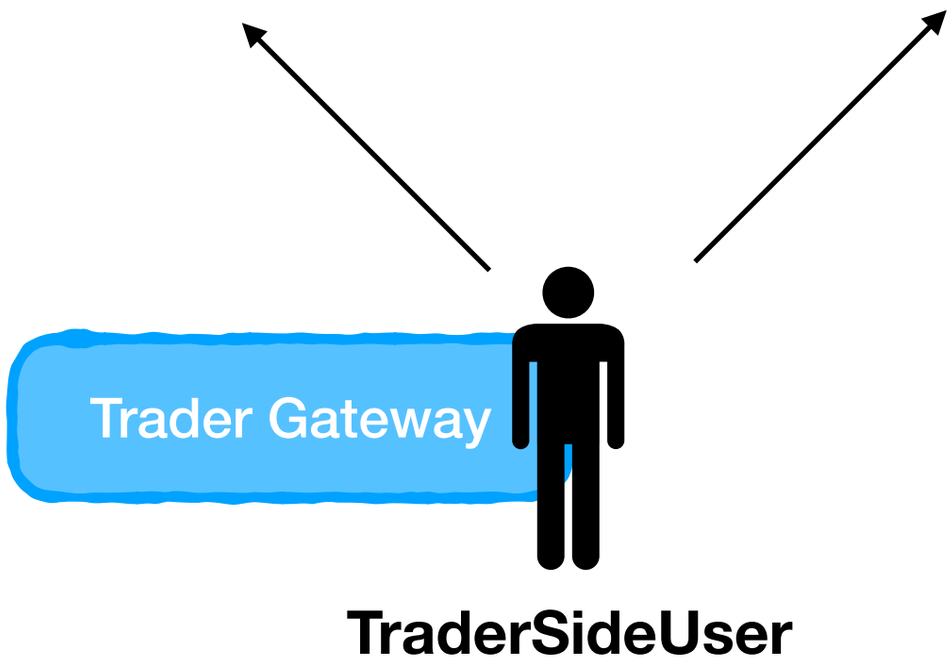
Redis for broker token



Trader Gateway

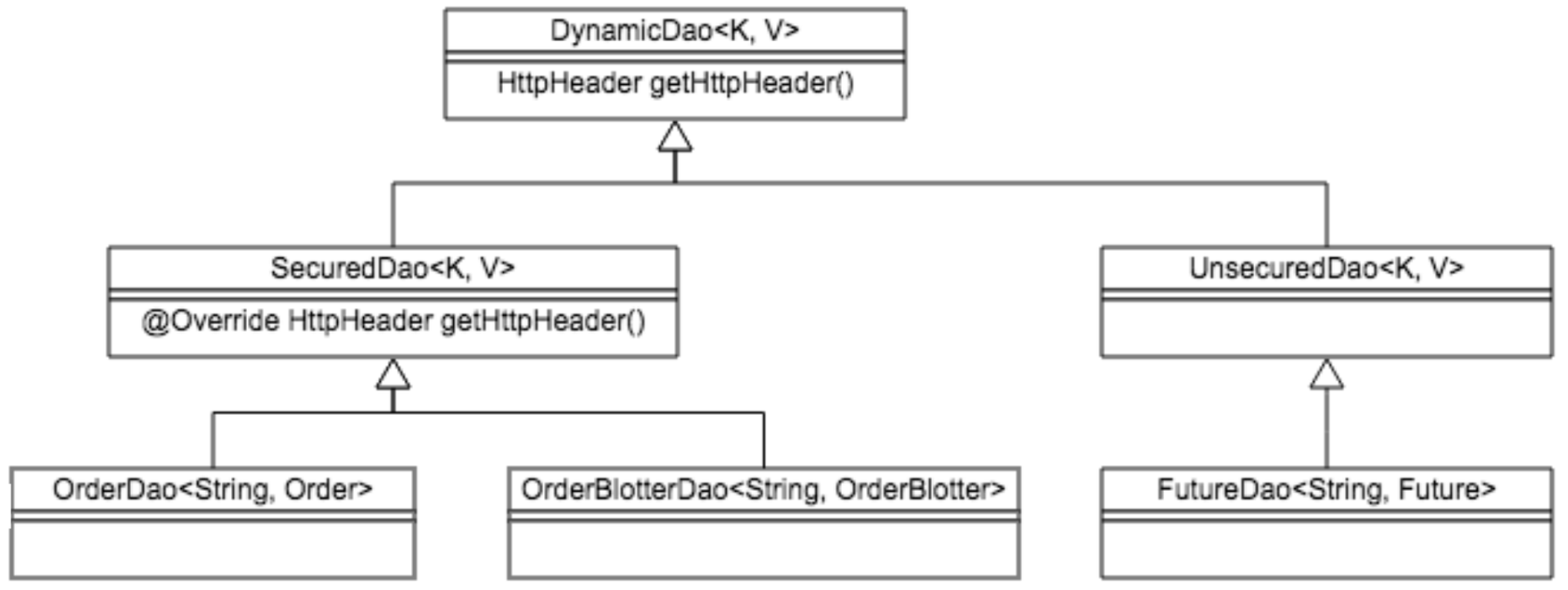


- Multiple Brokers



Trader Gateway

- Multiple Brokers



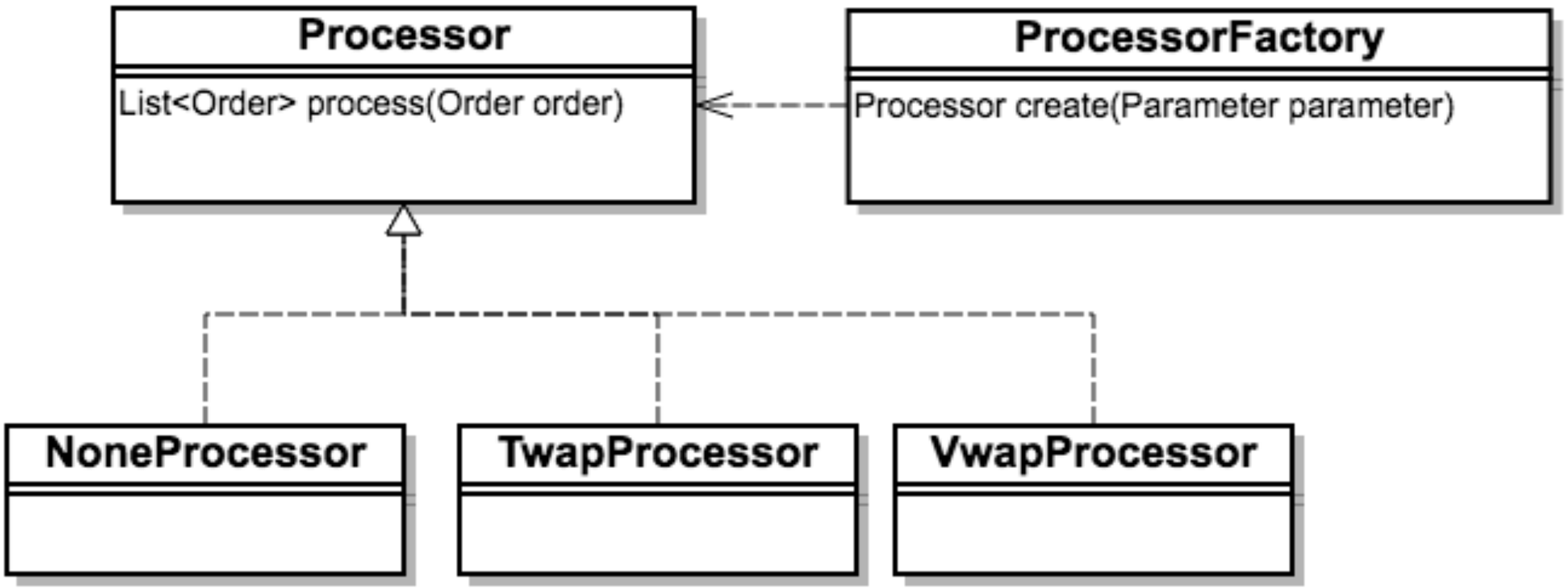
Trader Gateway

- Multiple Brokers
- Order
  - Phase 1 - Process
  - Phase 2 - Send

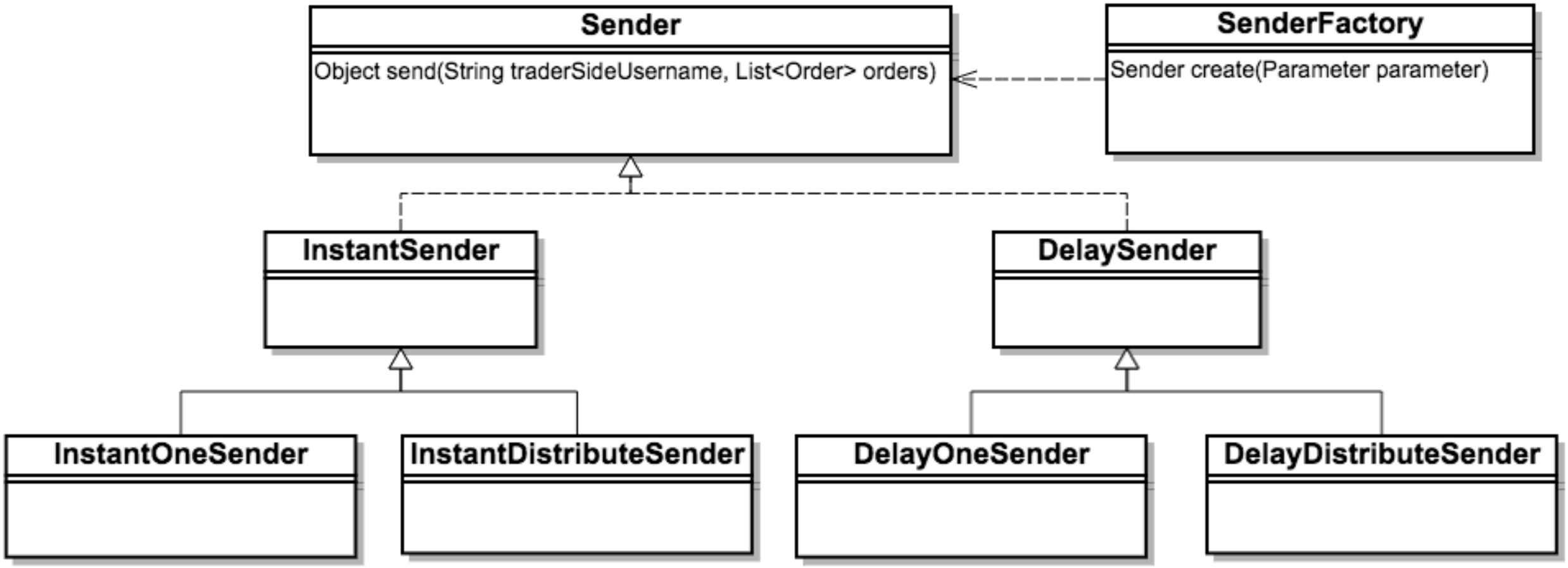
Trader Gateway

```
@Override
public Object createWithStrategy(String username, Order order,
                                  ProcessorFactory.Parameter pp,
                                  SenderFactory.Parameter sp){
    Processor processor = processorFactory.create(pp);
    List<Order> orders = processor.process(order);

    Sender sender = senderFactory.create(sp);
    Object res  = sender.send(username, orders);

    return res;
}
```

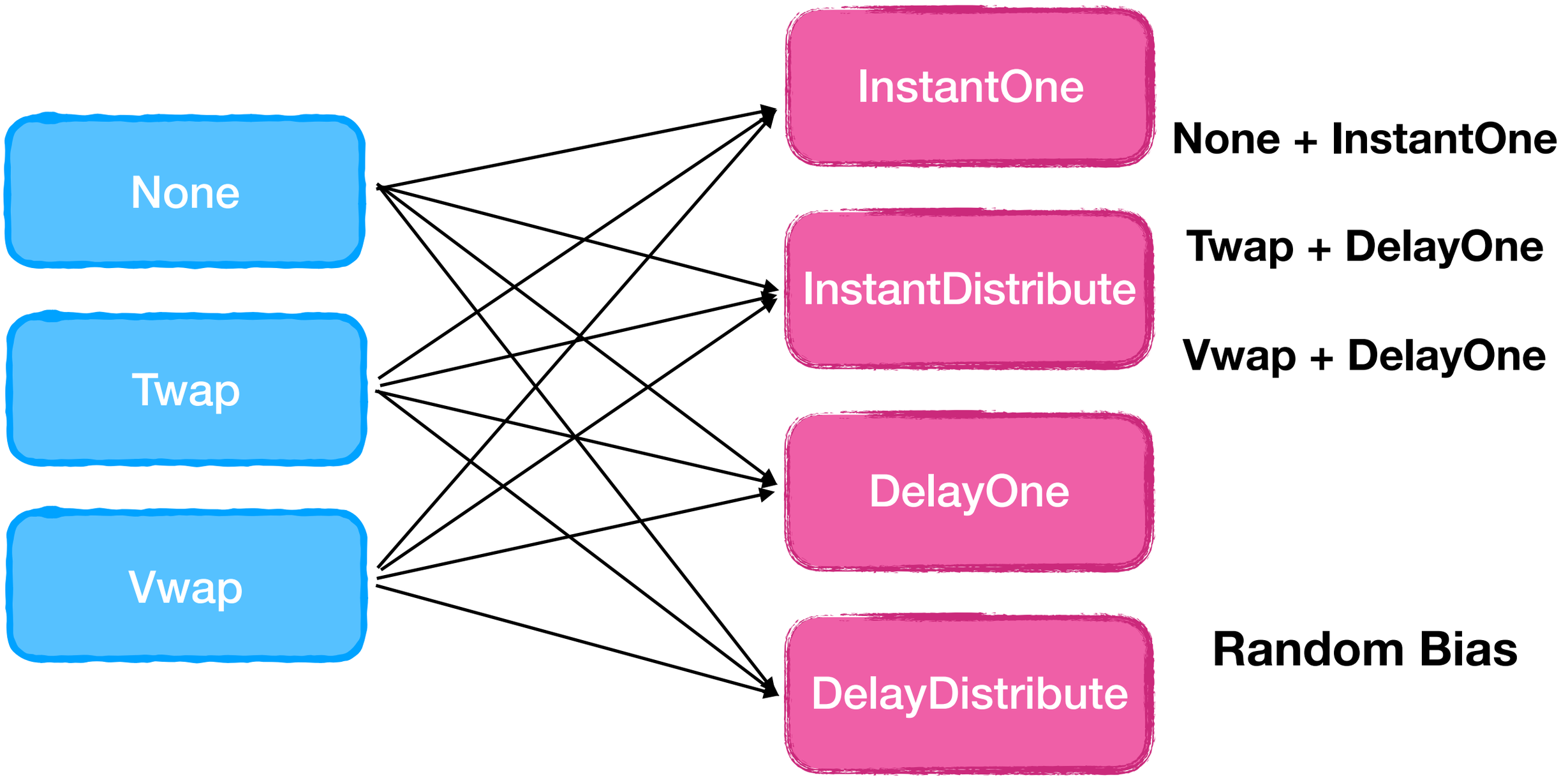
Trader Gateway



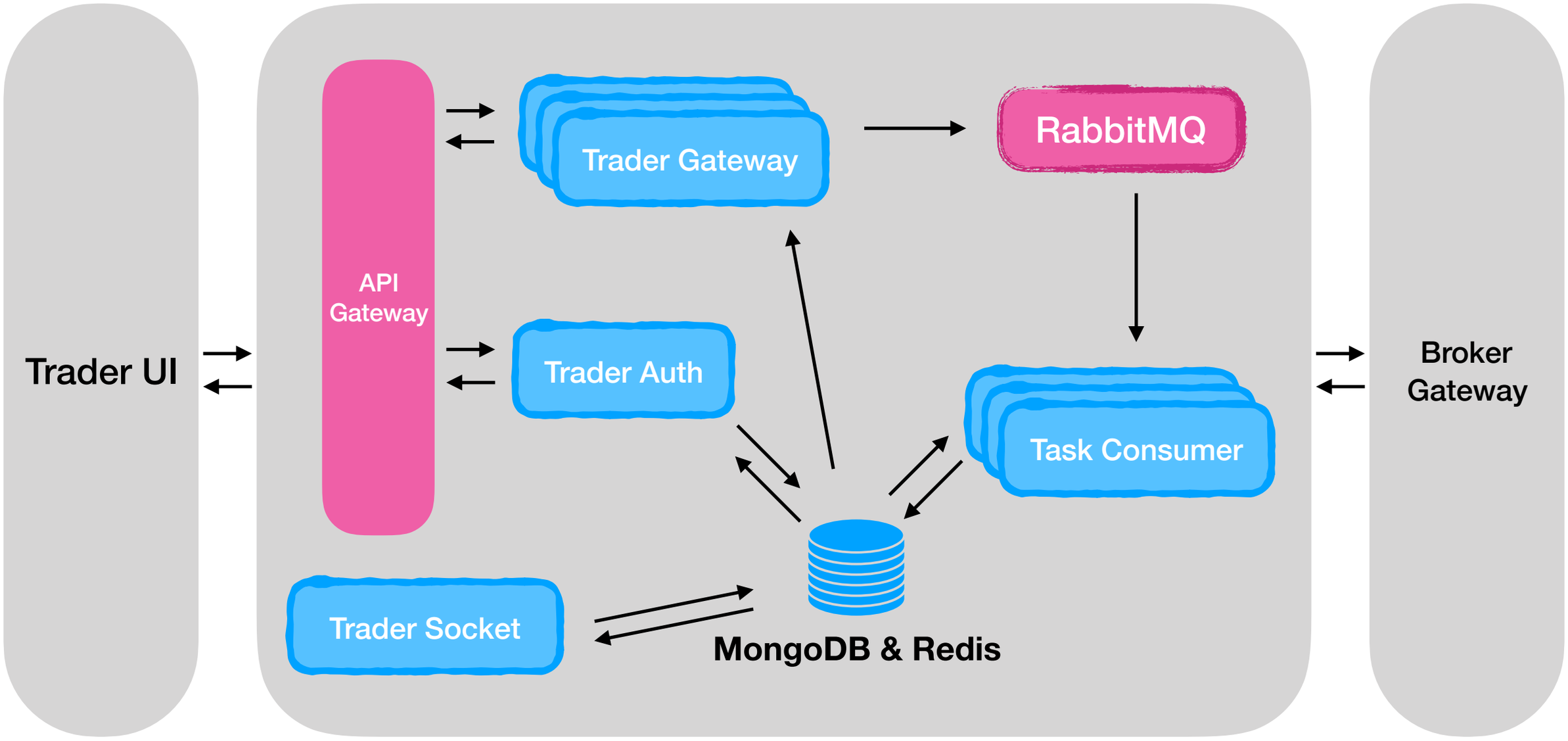
Trader Gateway



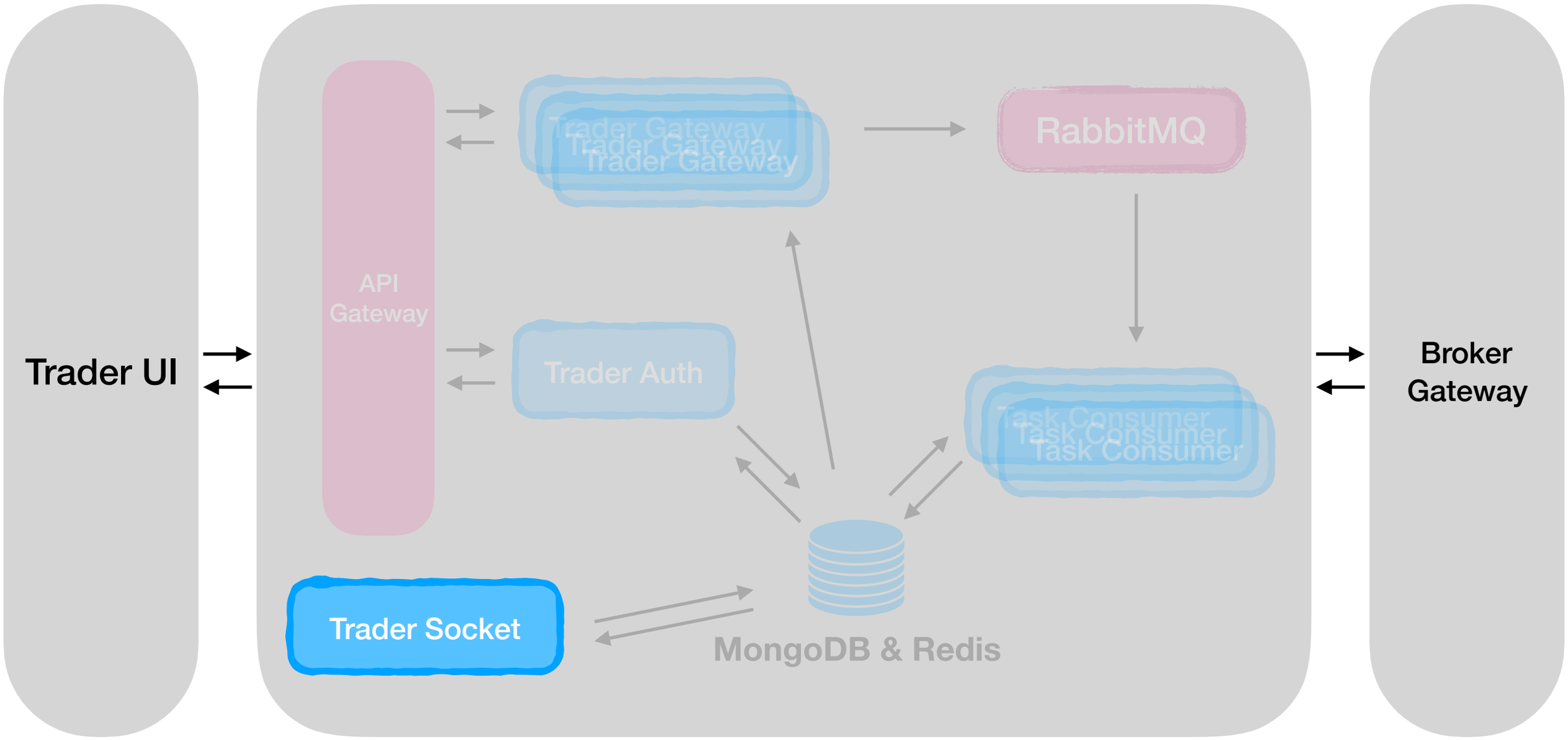
Trader Gateway



Trader Component

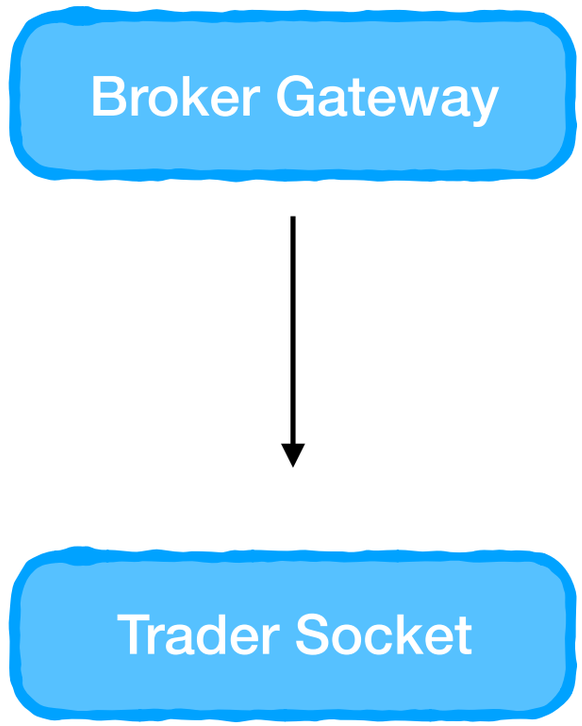


Trader Component



Trader Socket

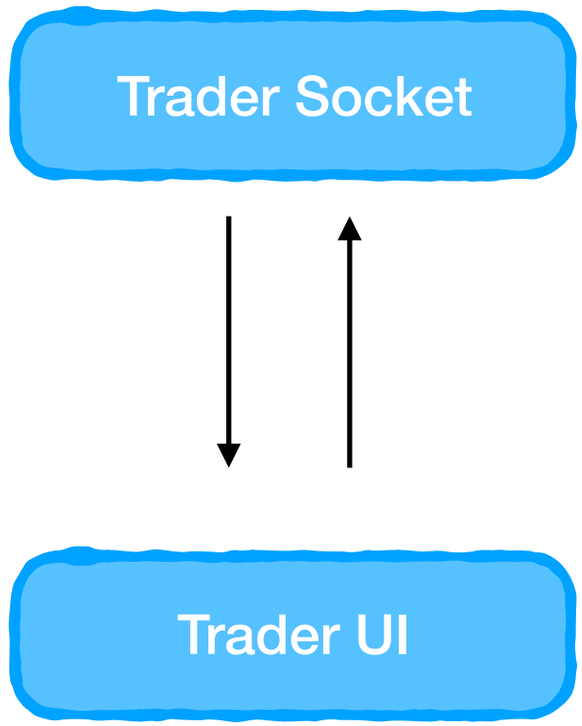
- Connect to Broker Gateway
    - Automatically reconnect



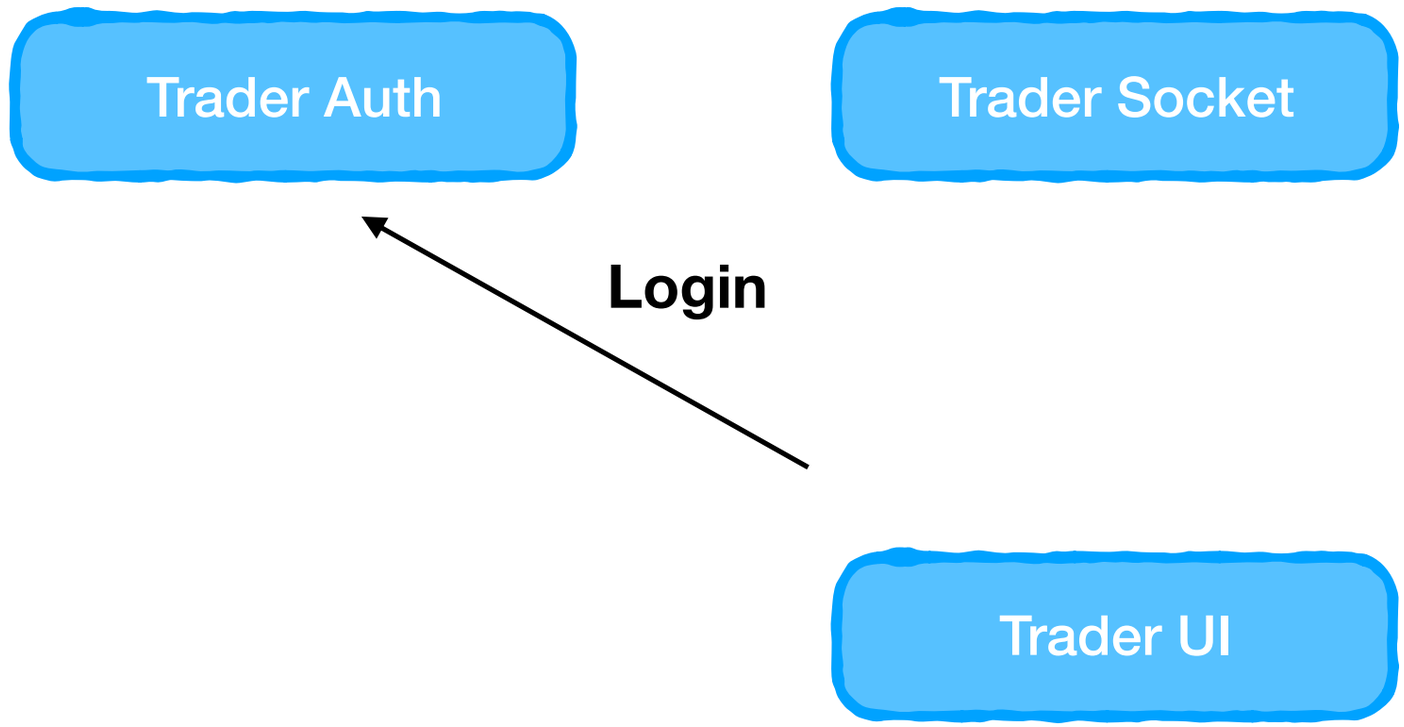
Process and store some data in memory

Trader Socket

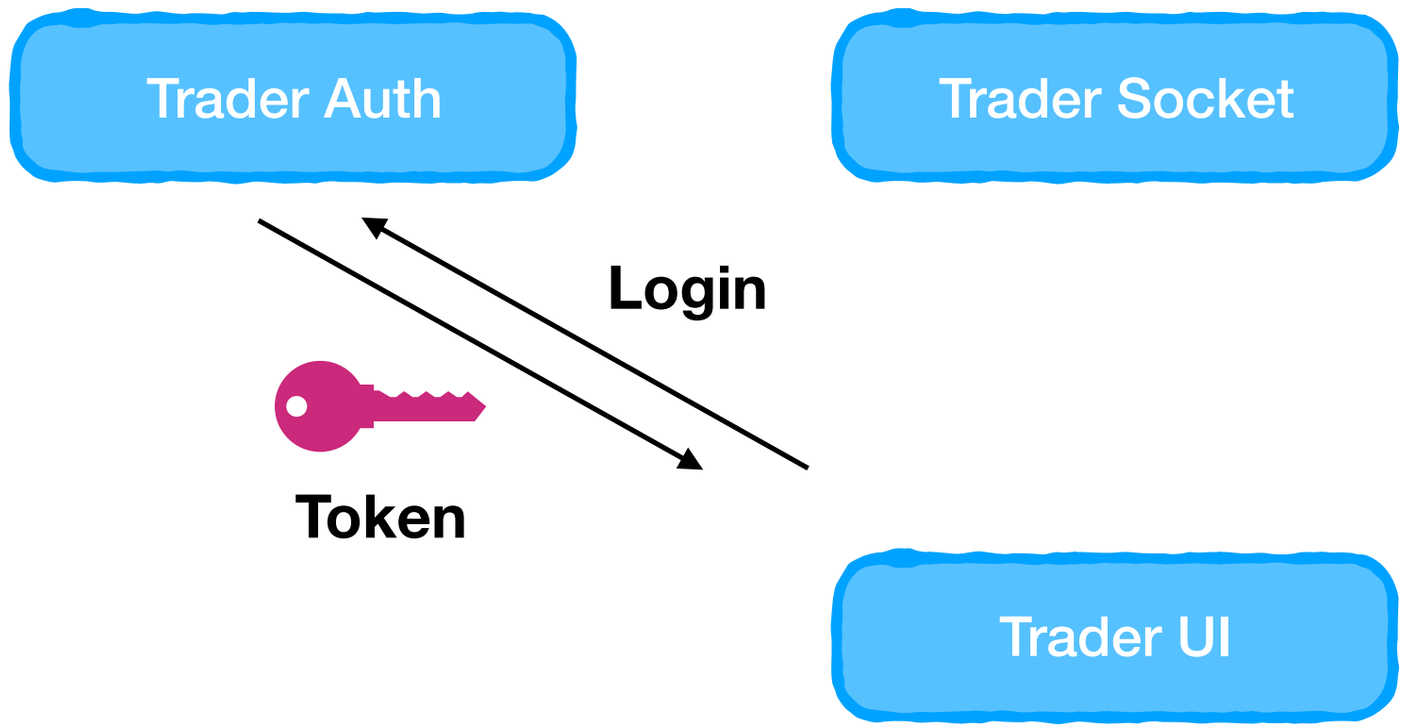
- Connect to Broker Gateway
  - Automatically reconnect
- Be connected by Trader UI
  - Login
  - Different brokers & futures
  - Only one socket for each user



Trader Socket



Trader Socket



Trader Socket

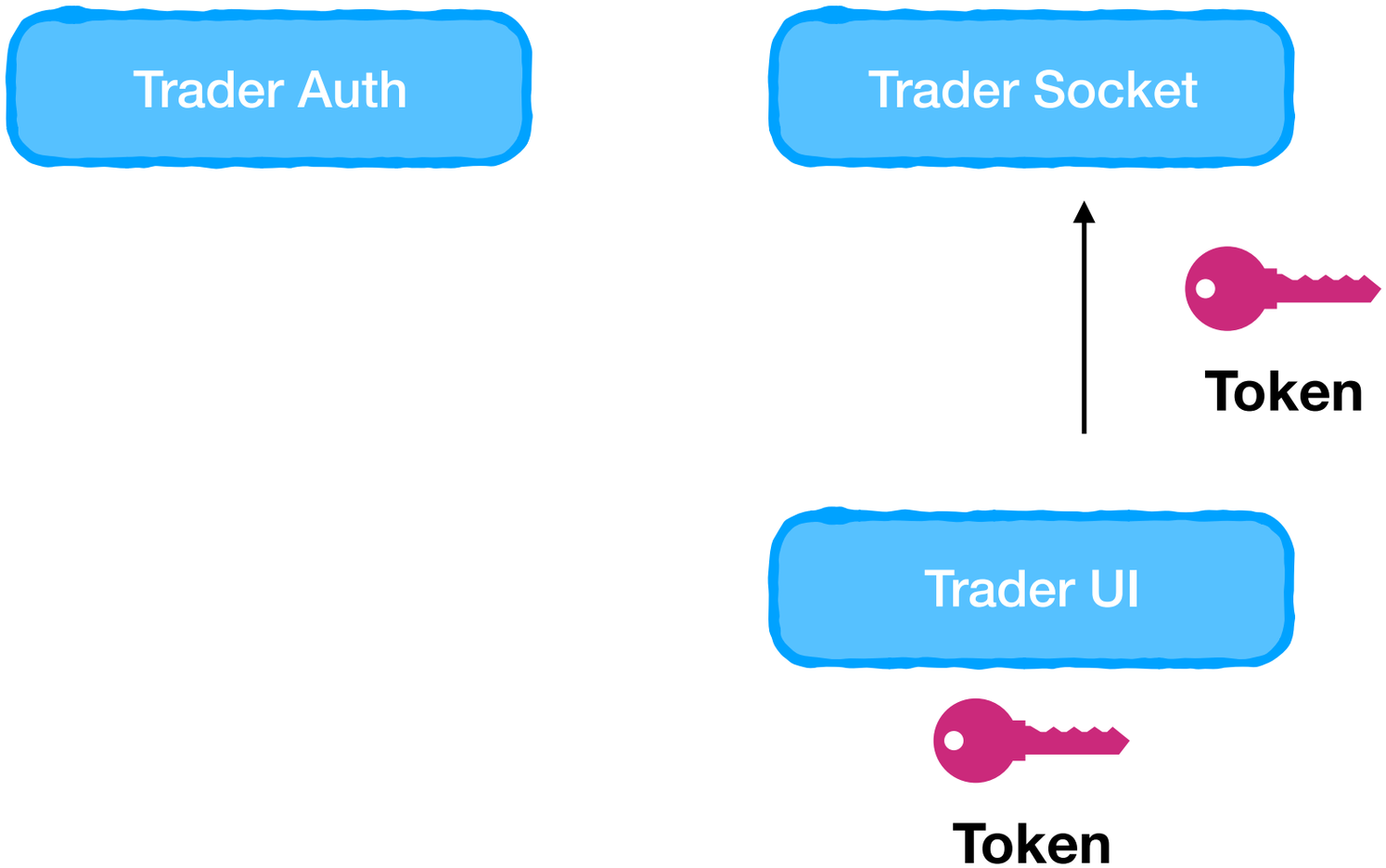
Trader Auth

Trader Socket

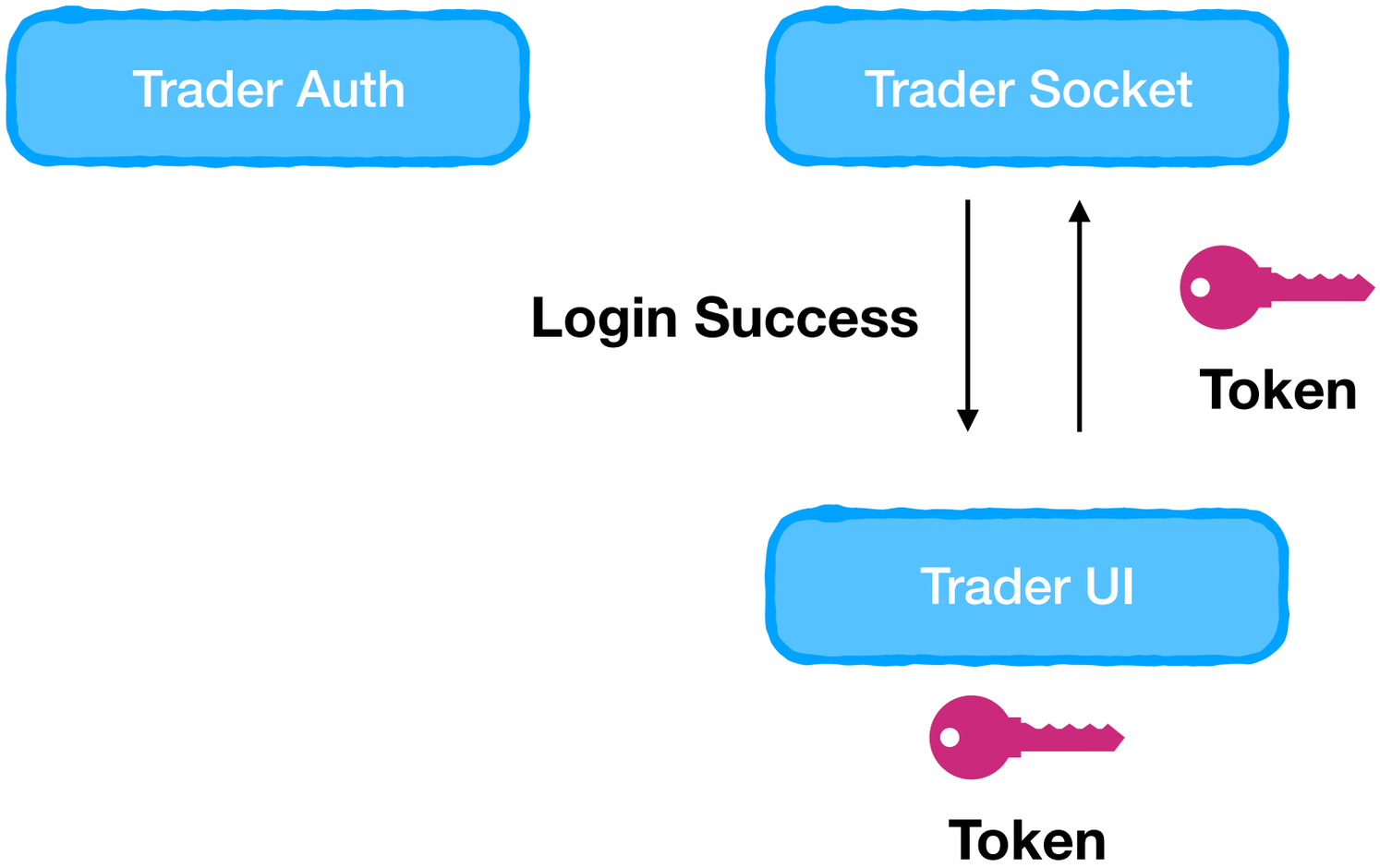
Trader UI

Token

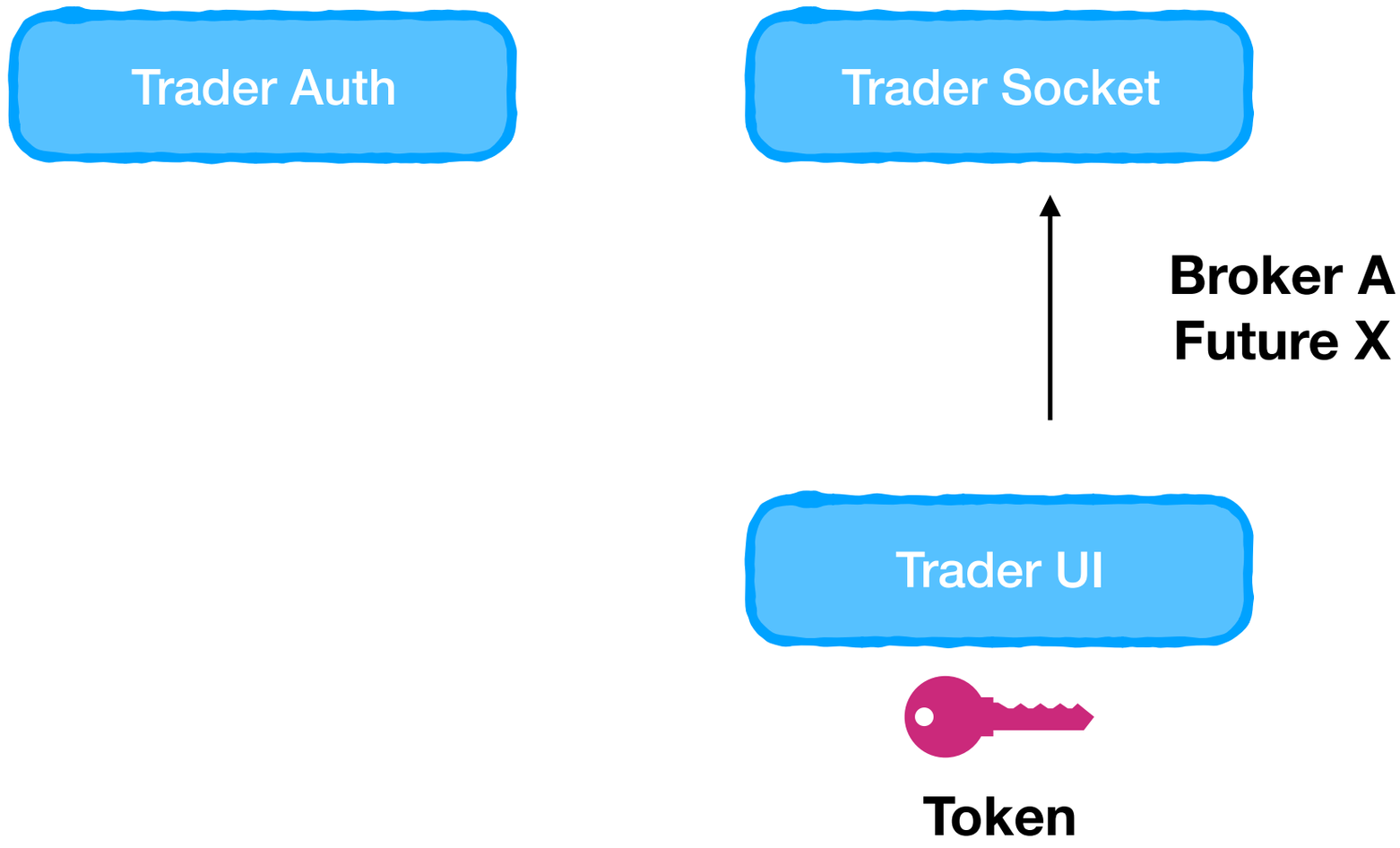
Trader Socket



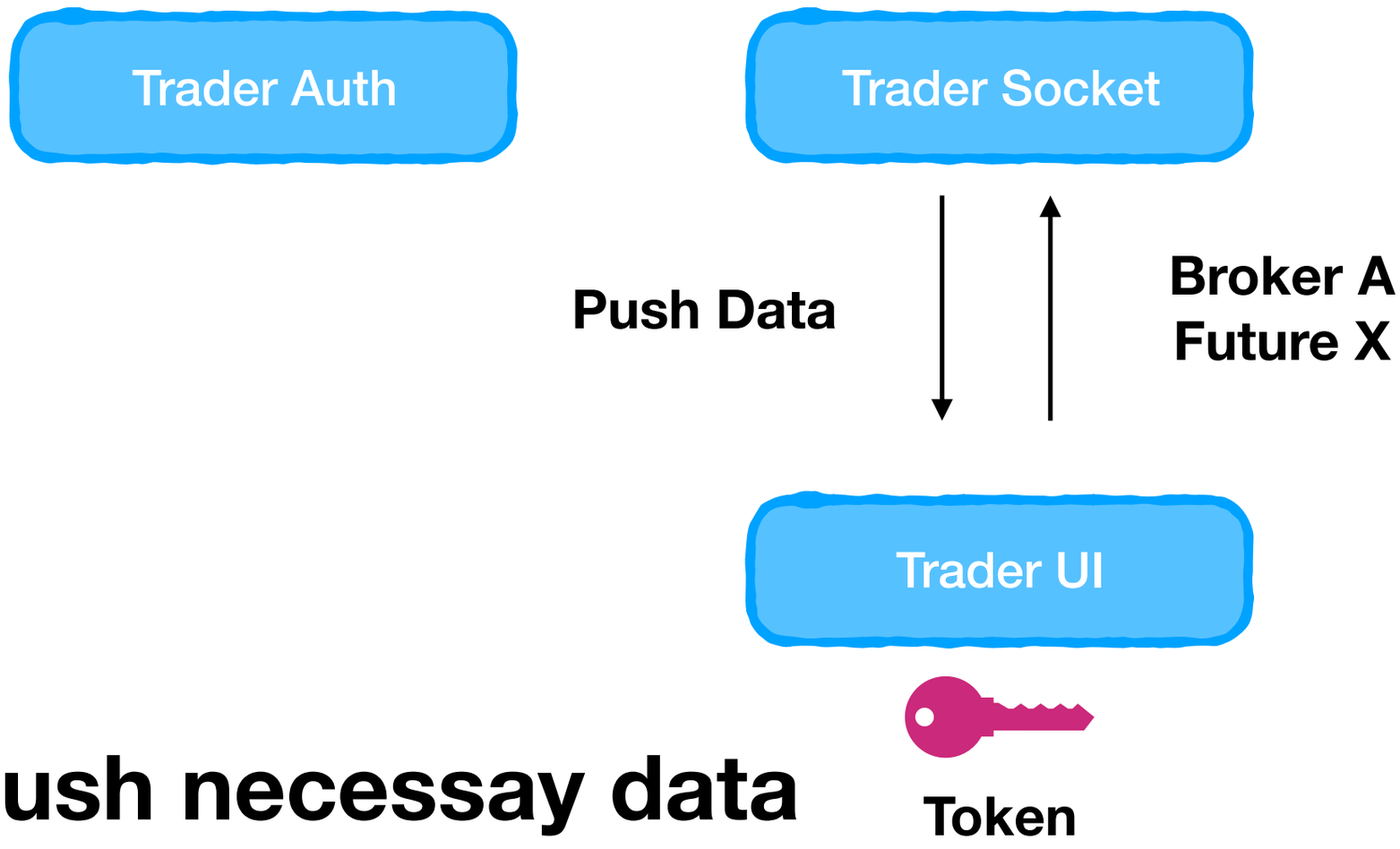
Trader Socket



Trader Socket



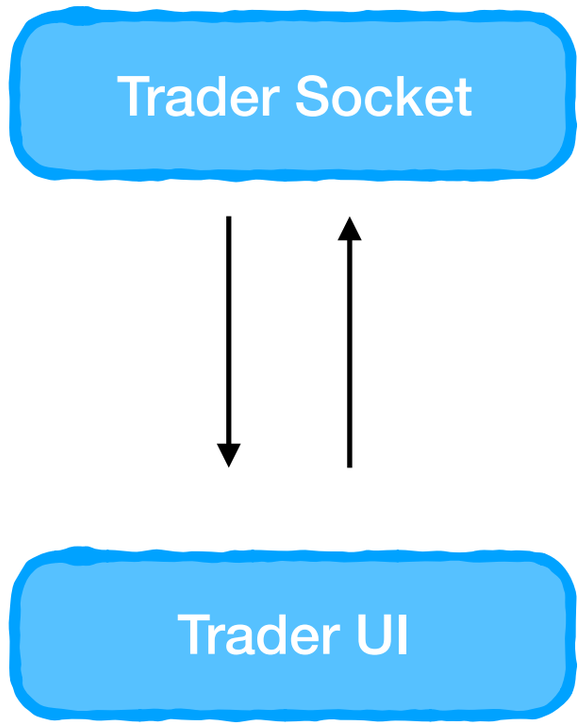
Trader Socket



Only push necessay data

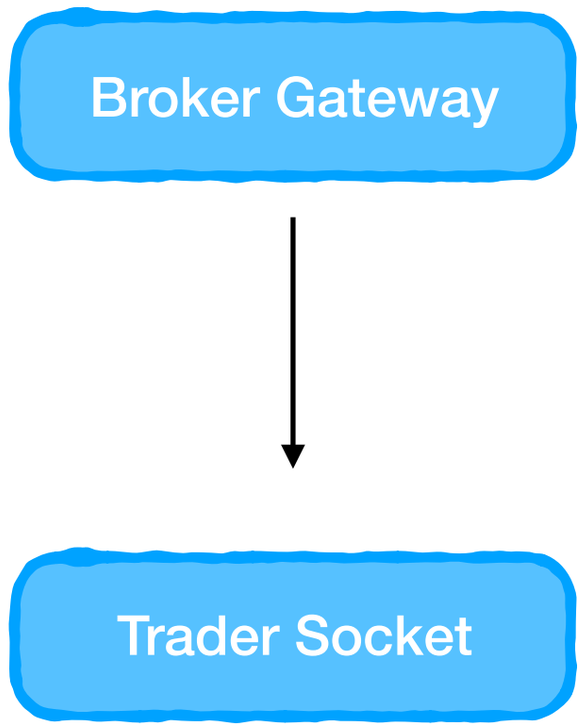
Trader Socket

```
{
    “type": “login",
    “body": {
        “token”: “xxx”,
        “marketDepthId": "xxx",
        “brokerId": "xxx"
    }
}

{
    “type": "switch",
    “body": {
        “marketDepthId": "xxx",
        “brokerId": "xxx"
    }
}
```



Trader Socket

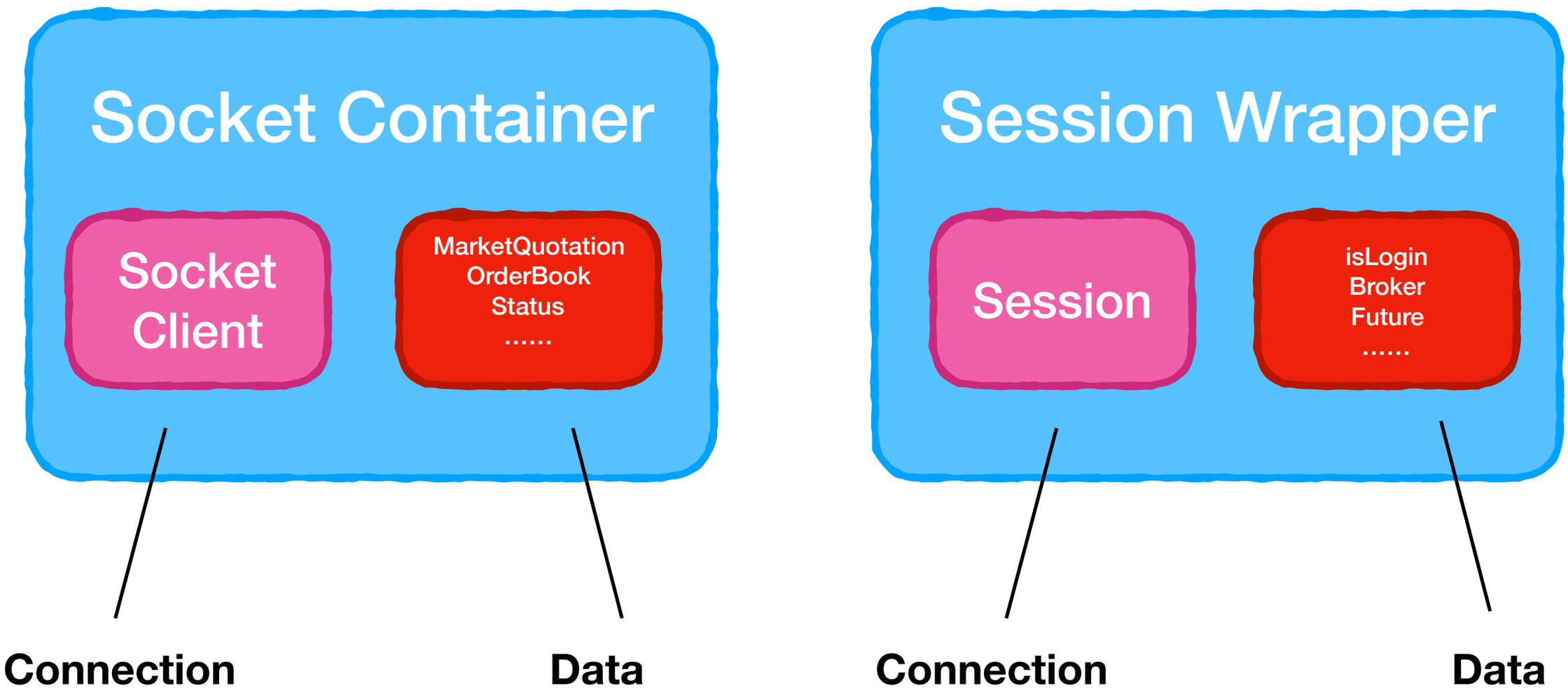
- Connect to Broker Gateway
  - Automatically reconnect
- Be connected by Trader UI
  - Login
  - Different brokers & futures
  - Only one socket for each user
- Broker Management



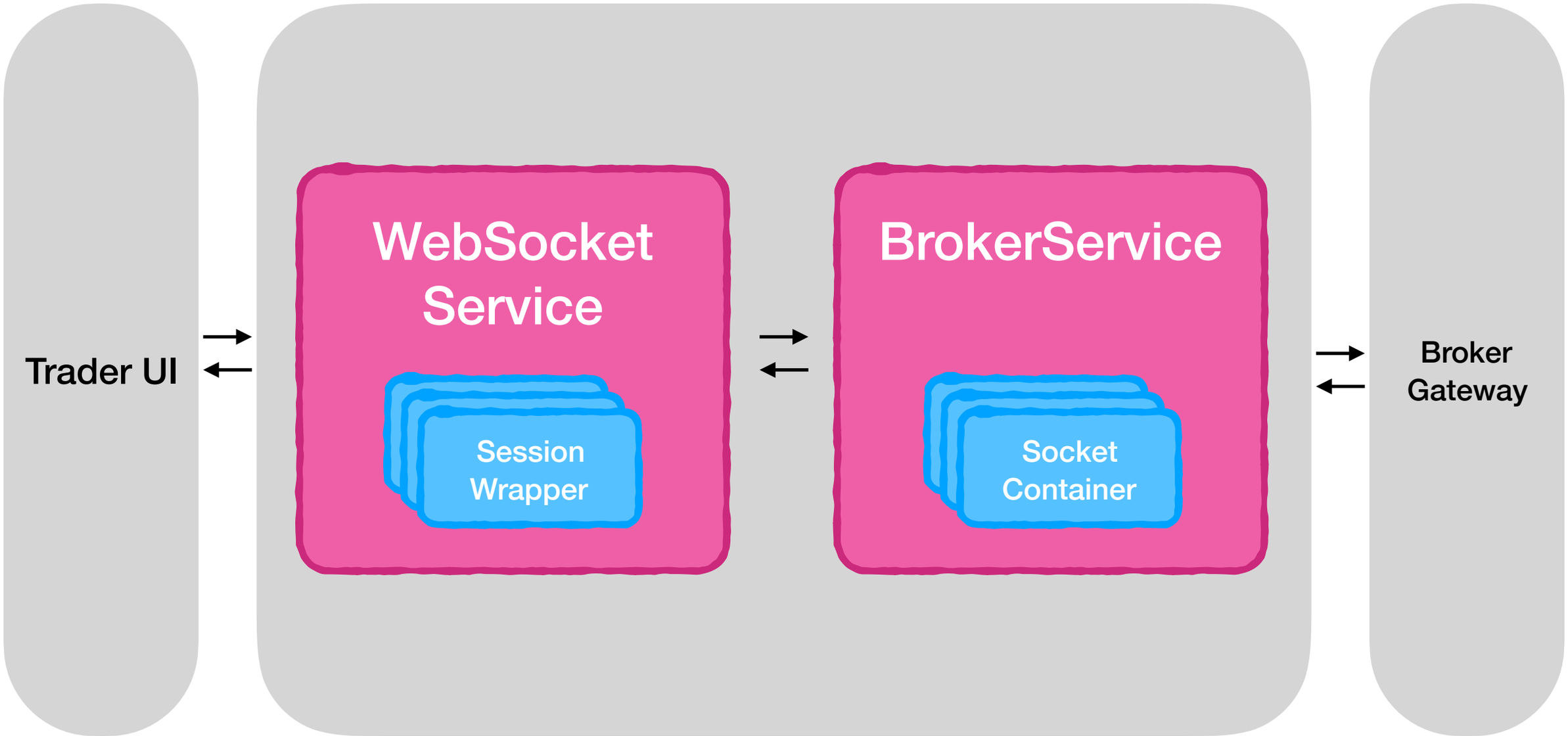
Trader Socket

Broker Gateway

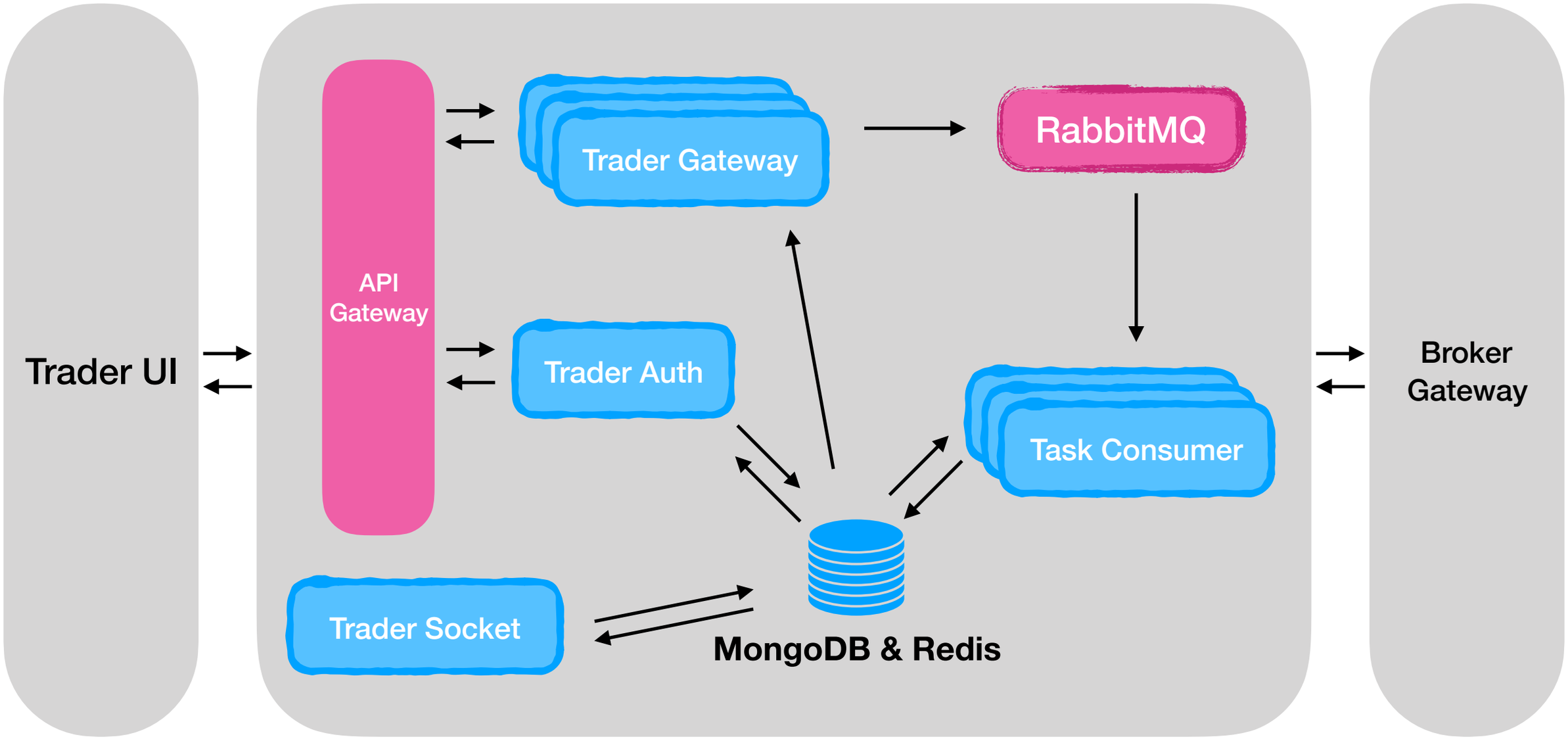
Trader UI



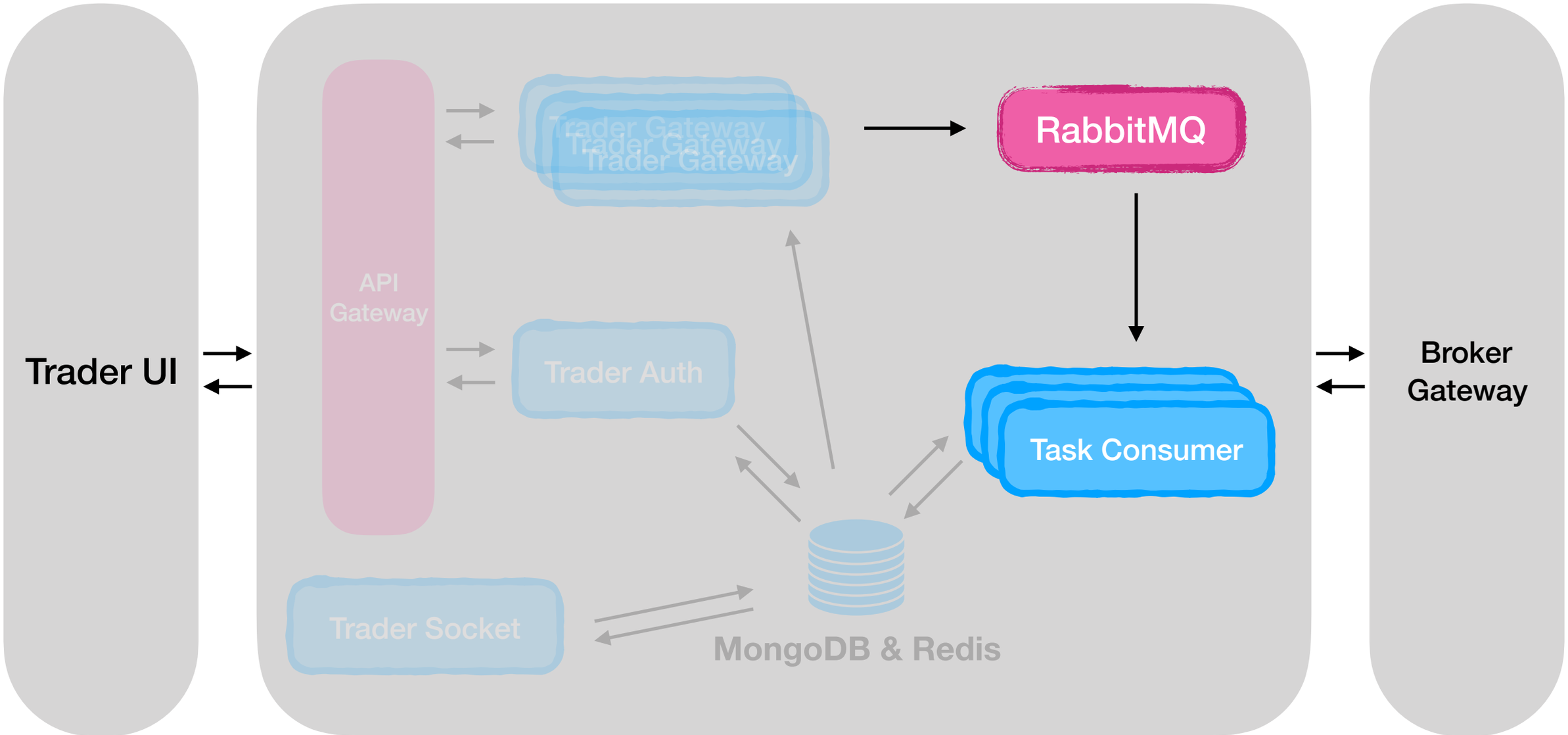
Trader Socket



Trader Component



Trader Component

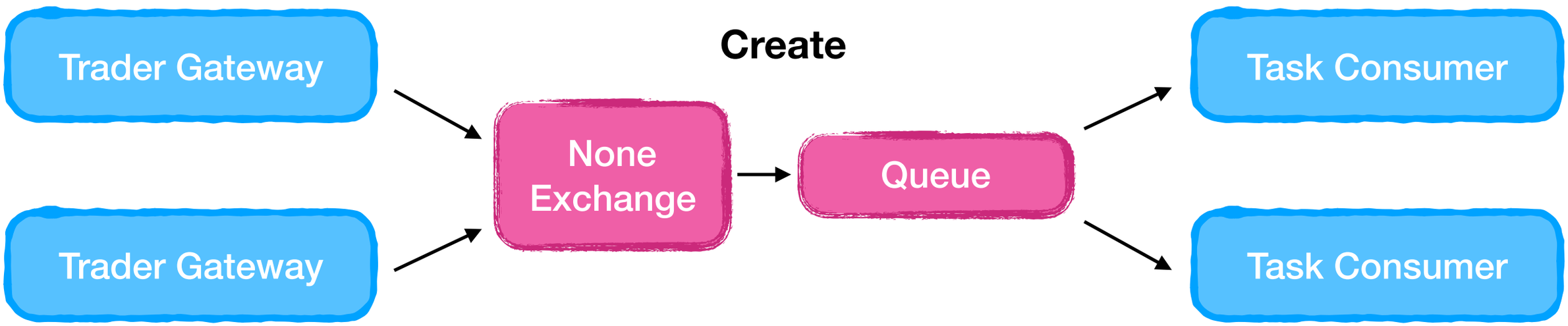


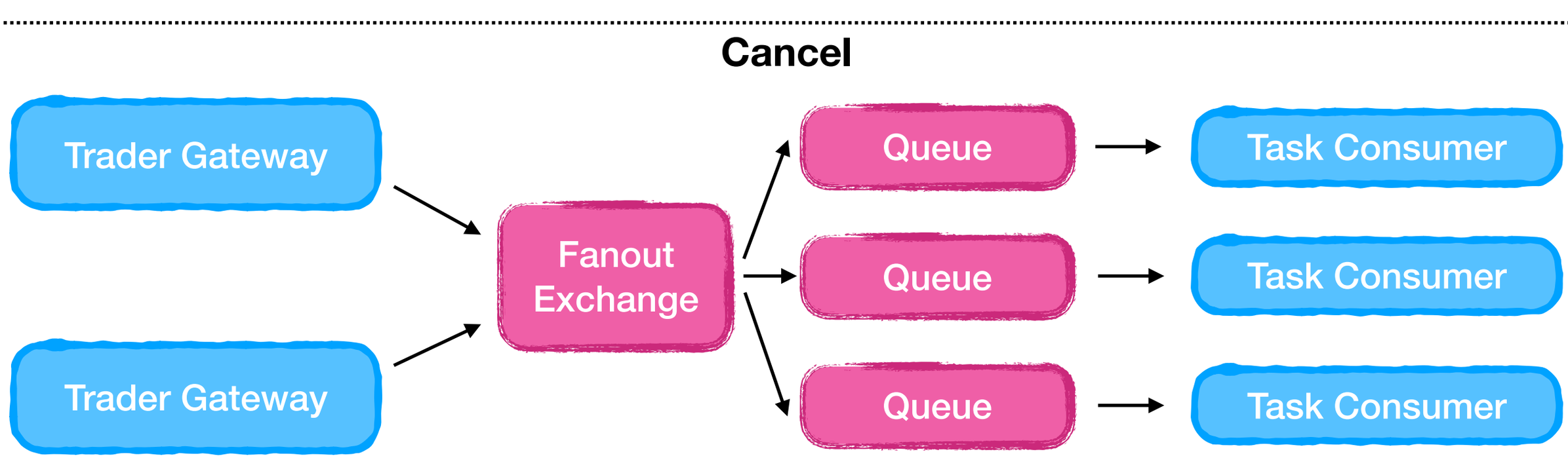
Task Consumer

- DelaySender sends request to RabbitMQ

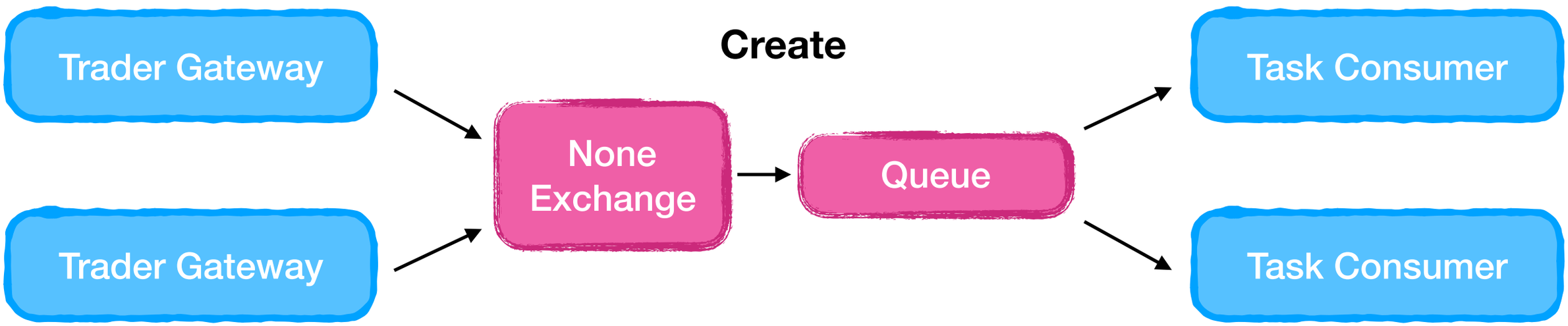
MongoDB & Redis

Task Consumer





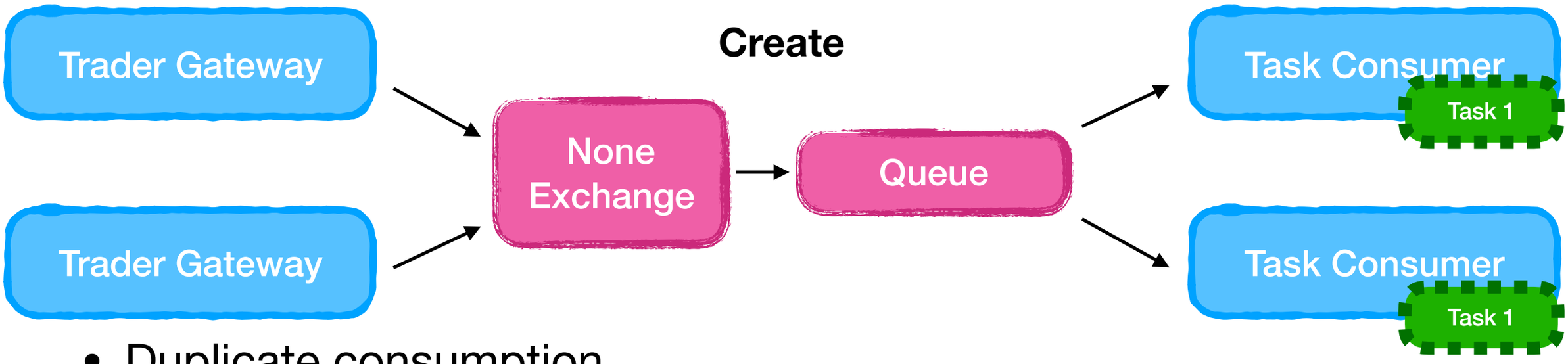
Task Consumer



- Duplicate consumption
  - Distributed lock - Redis
  - Check ID

```
if redis.exist(key):
    return
getLock = redis.setIfNotExist(key, 0)
if getLock:
    #consume
else:
    return
```

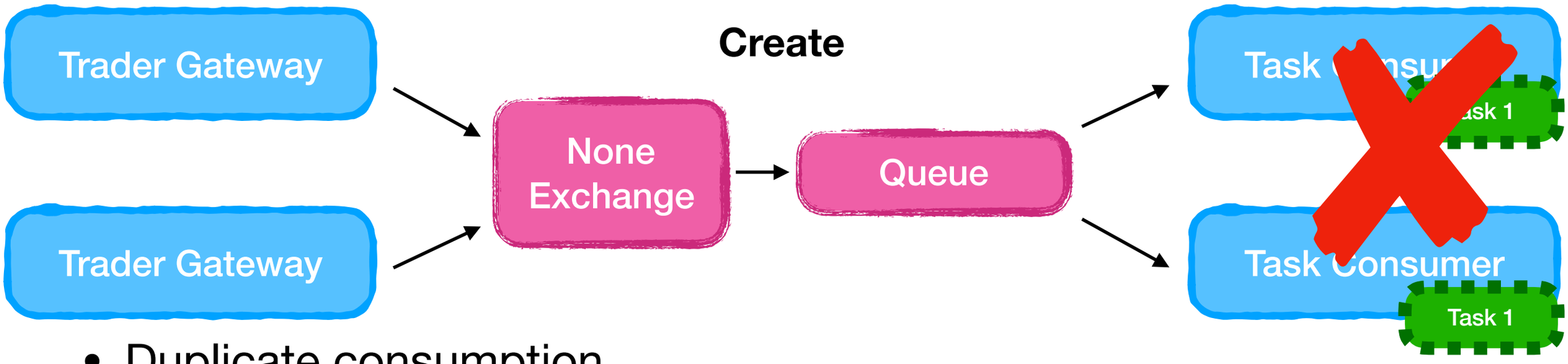
Task Consumer



- Duplicate consumption
- Distributed lock - Redis
- Check ID

```
if redis.exist(key):
    return
getLock = redis.setIfNotExist(key, 0)
if getLock:
    #consume
else:
    return
```

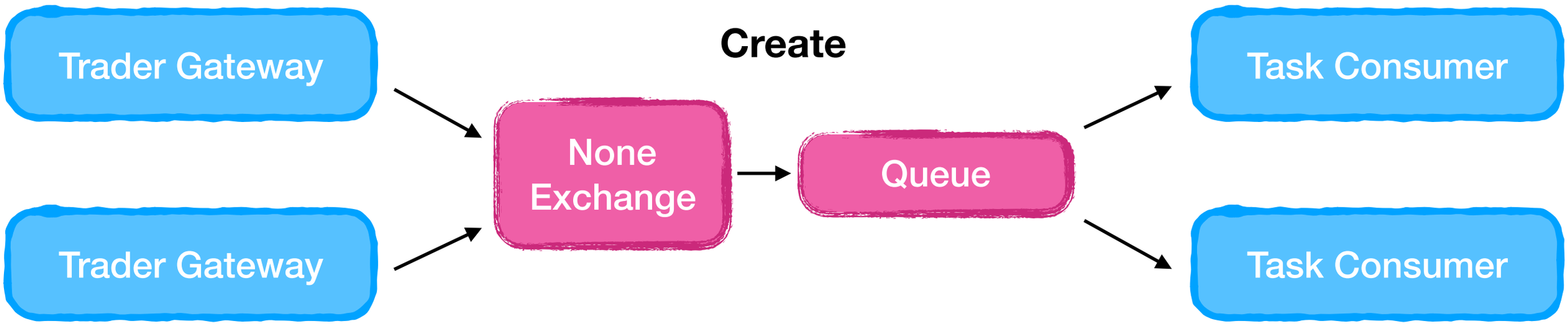
Task Consumer



- Duplicate consumption
- Distributed lock - Redis
- Check ID

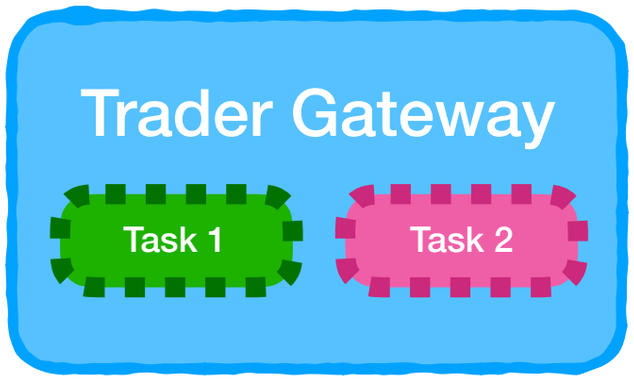
```
if redis.exist(key):
    return
getLock = redis.setIfNotExist(key, 0)
if getLock:
    #consume
else:
    return
```

Task Consumer



- Duplicate consumption
  - Distributed lock - Redis
  - Check ID
- Message lost
  - AutoAck = false
  - => replica is ok

Without Message Queue



- Duplicate consumption
  - Distributed lock - Redis
  - Check ID
- Message lost
  - AutoAck = false
  - => replica is ok

Without Message Queue

- Duplicate consumption
  - Distributed lock - Redis
  - Check ID
- Message lost
  - AutoAck = false
  - => replica is ok

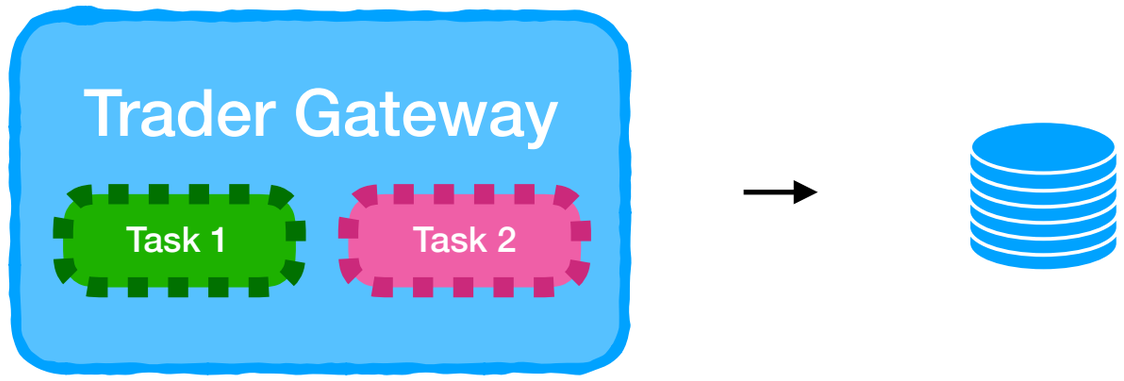
Without Message Queue

- Duplicate consumption
  - Distributed lock - Redis
  - Check ID

Task 1 and Task 2 are lost !!!

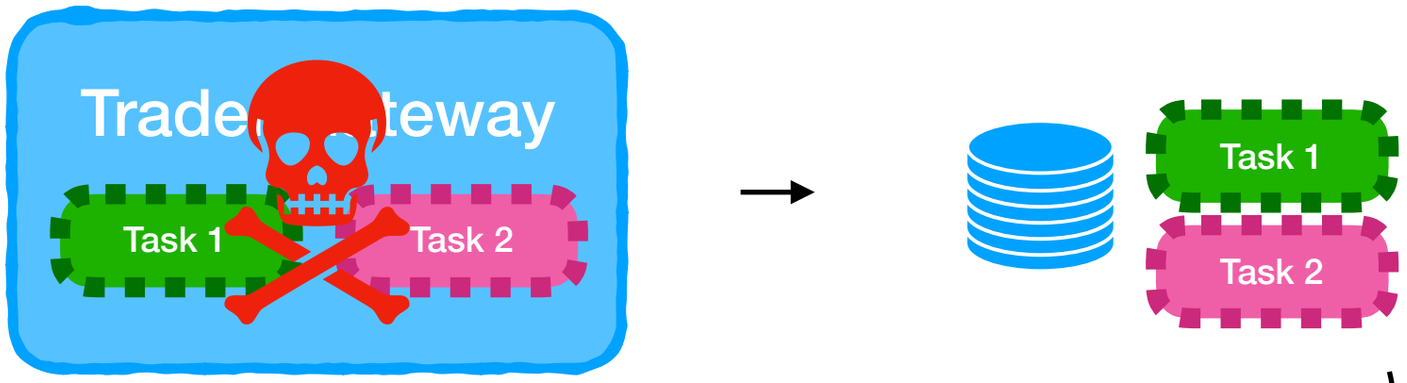
- Message lost
  - AutoAck = false
  - => replica is ok

Bad Solution



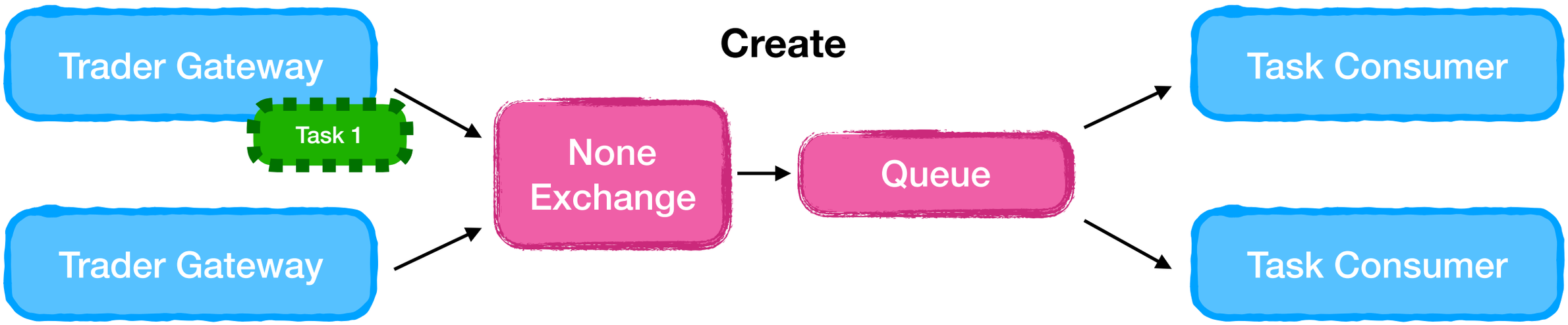
- Duplicate consumption
  - Distributed lock - Redis
  - Check ID
- Message lost
  - AutoAck = false
  - => replica is ok

Bad Solution



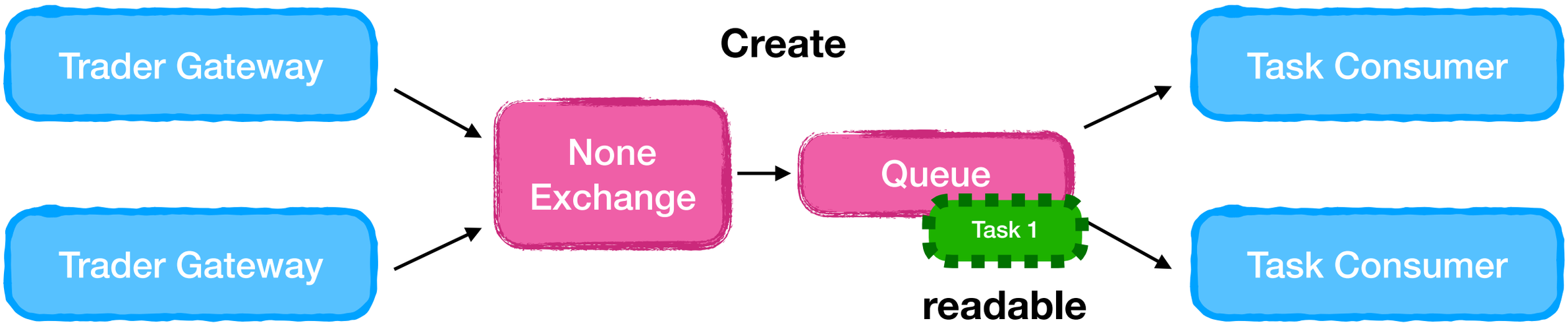
- Duplicate consumption
  - Distributed lock - Redis
  - Check ID
- Message lost
  - AutoAck = false
  - => replica is ok

Situation 1



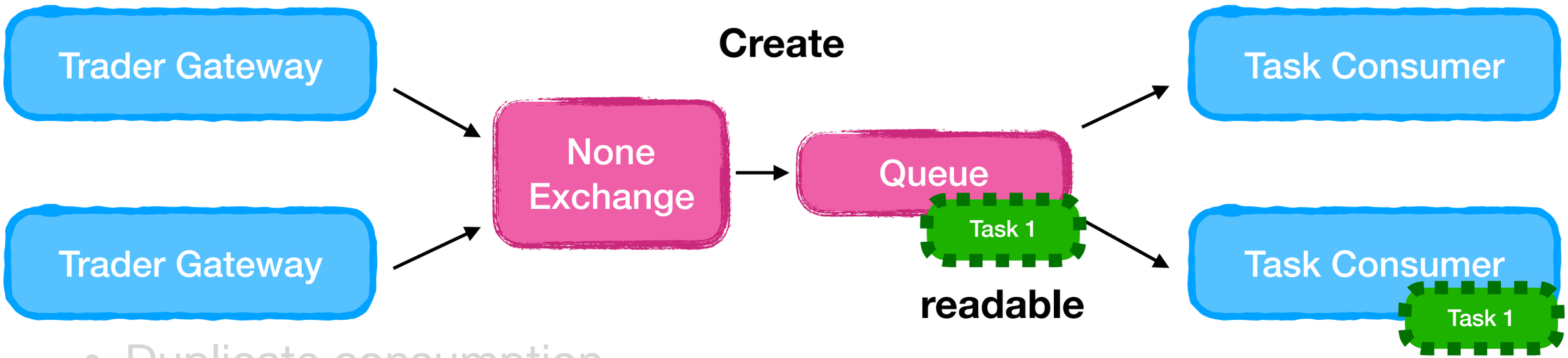
- Duplicate consumption
  - Distributed lock - Redis
  - Check ID
- Message lost
  - AutoAck = false
  - => replica is ok

Situation 1



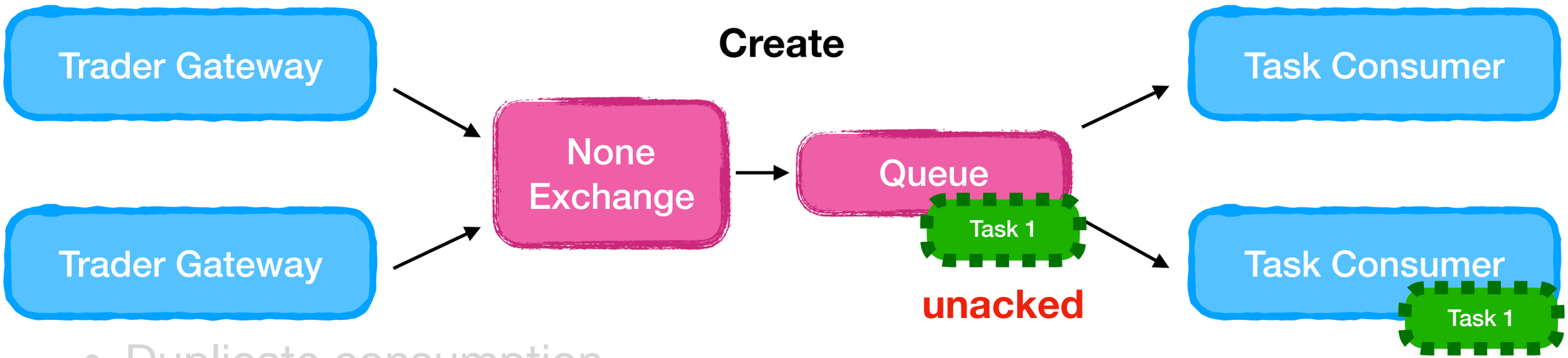
- Duplicate consumption
  - Distributed lock - Redis
  - Check ID
- Message lost
  - AutoAck = false
  - => replica is ok

Situation 1



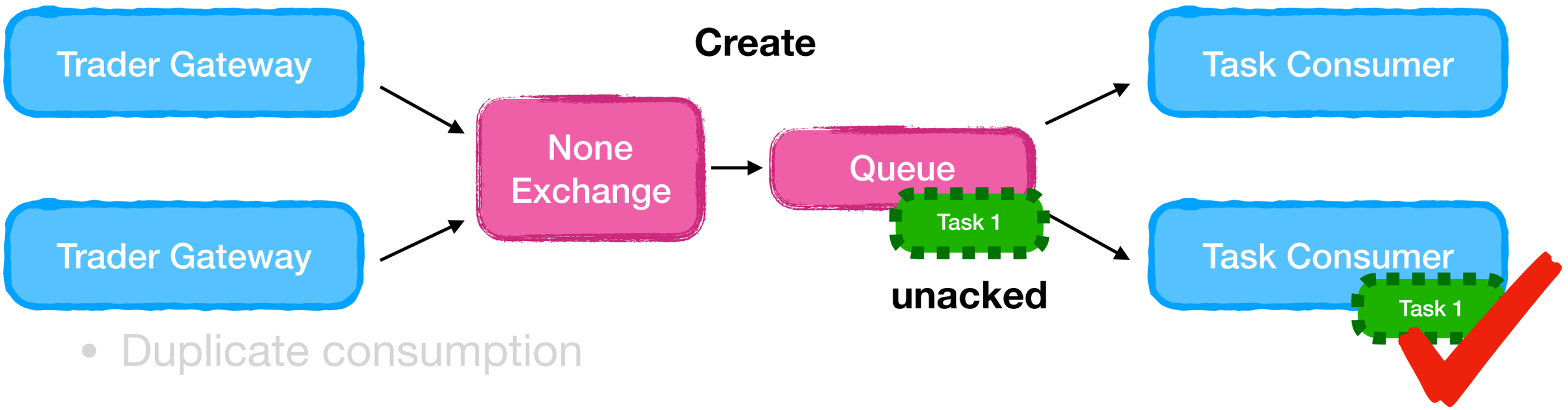
- Duplicate consumption
  - Distributed lock - Redis
  - Check ID
- Message lost
  - AutoAck = false
  - => replica is ok

Situation 1



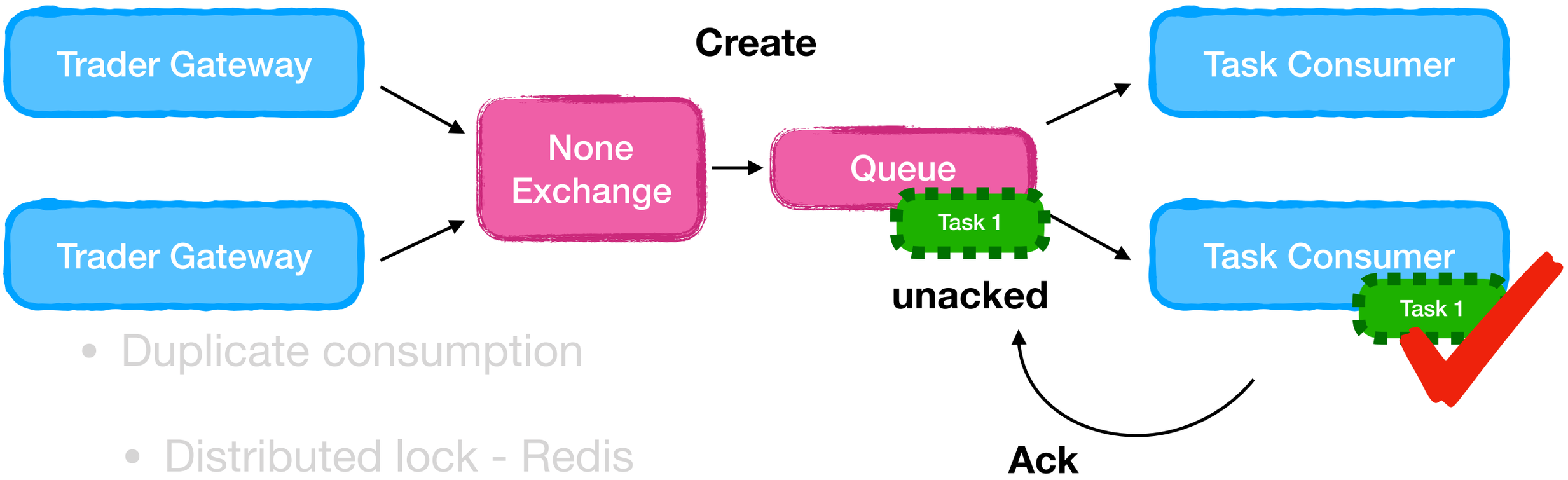
- Duplicate consumption
  - Distributed lock - Redis
  - Check ID
- Message lost
  - AutoAck = false
  - => replica is ok

Situation 1



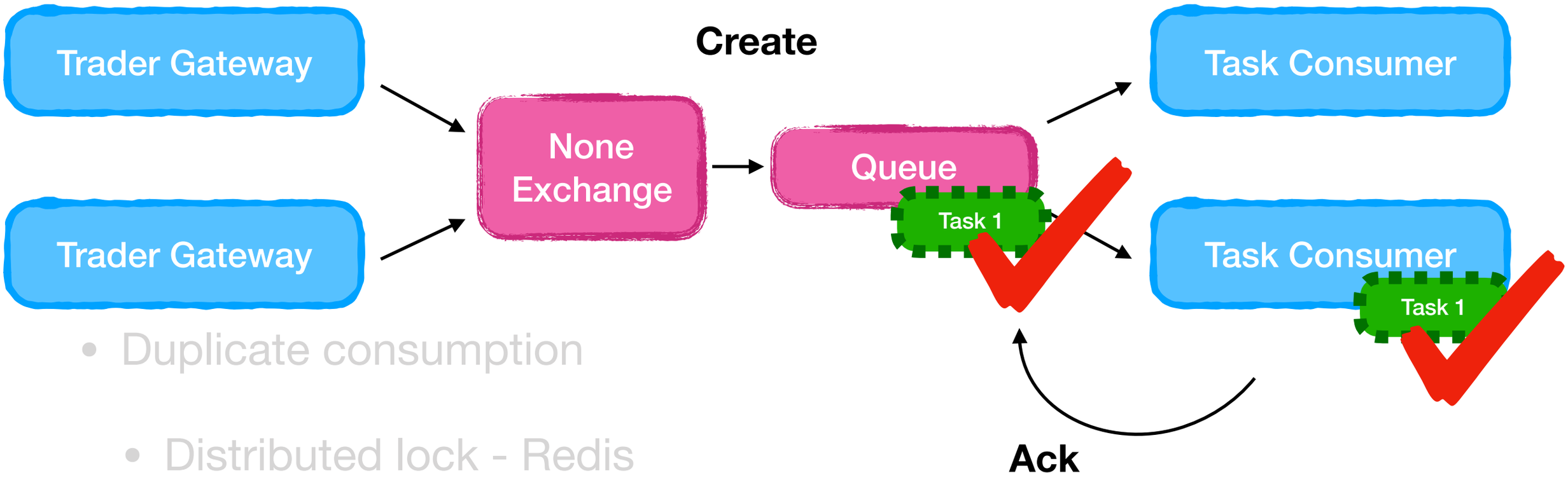
- Duplicate consumption
  - Distributed lock - Redis
  - Check ID
- Message lost
  - AutoAck = false
  - => replica is ok

Situation 1



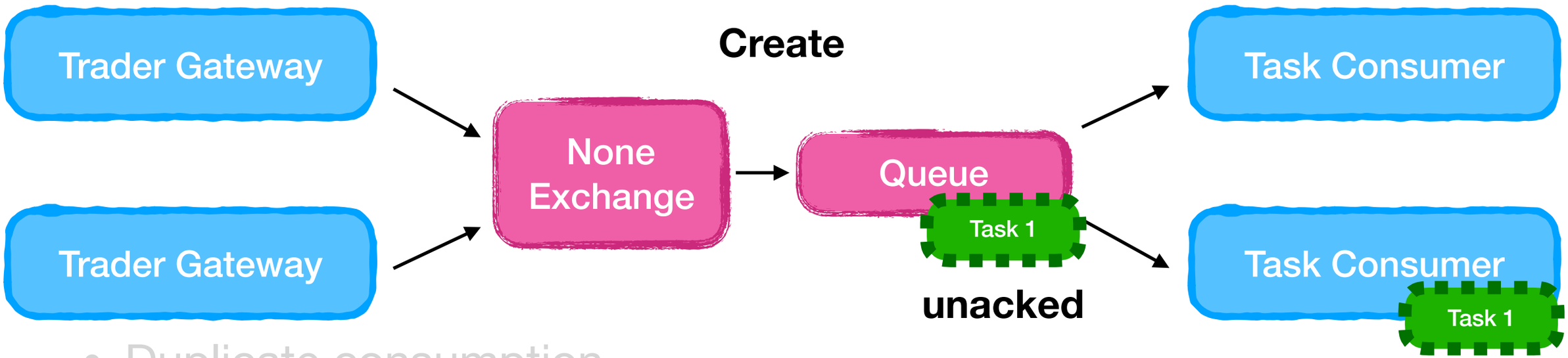
- Duplicate consumption
  - Distributed lock - Redis
  - Check ID
- Message lost
  - AutoAck = false
  - => replica is ok

Situation 1



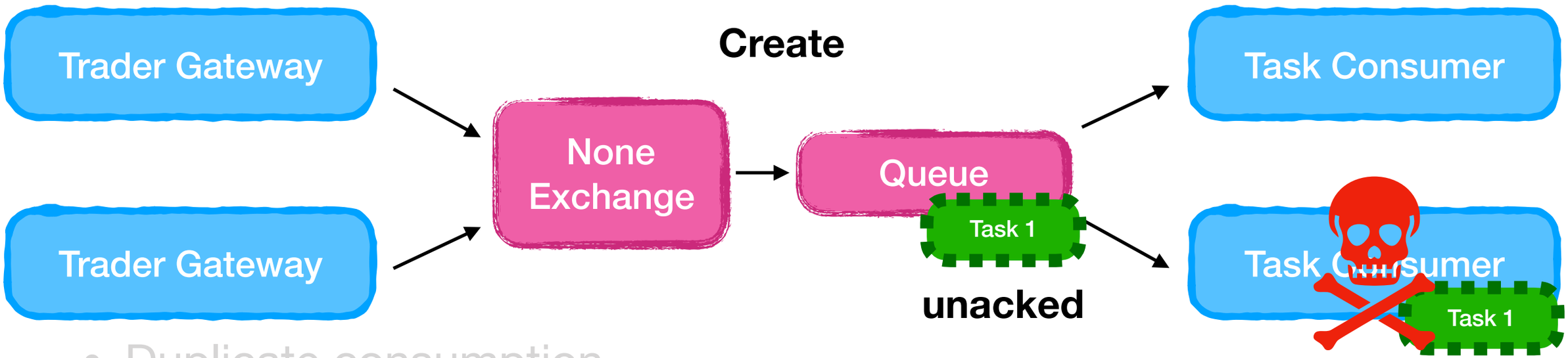
- Duplicate consumption
  - Distributed lock - Redis
  - Check ID
- Message lost
  - AutoAck = false
  - => replica is ok

Situation 2



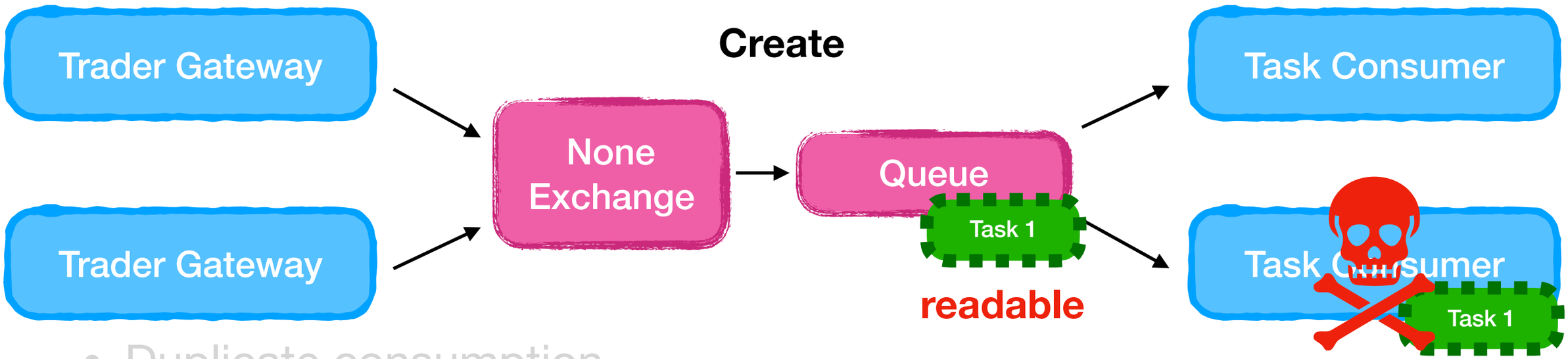
- Duplicate consumption
  - Distributed lock - Redis
  - Check ID
- Message lost
  - AutoAck = false
  - => replica is ok

Situation 2



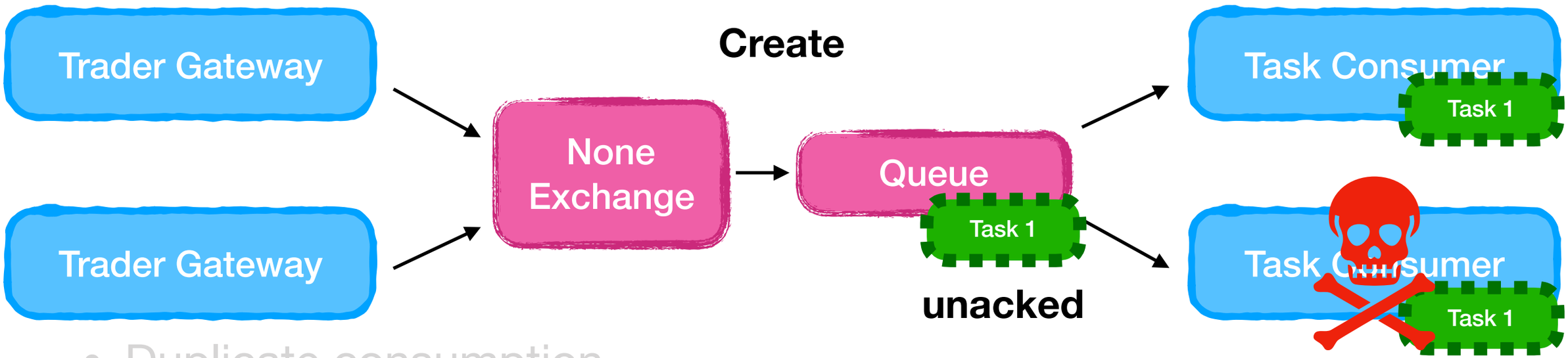
- Duplicate consumption
  - Distributed lock - Redis
  - Check ID
- Message lost
  - AutoAck = false
  - => replica is ok

Situation 2



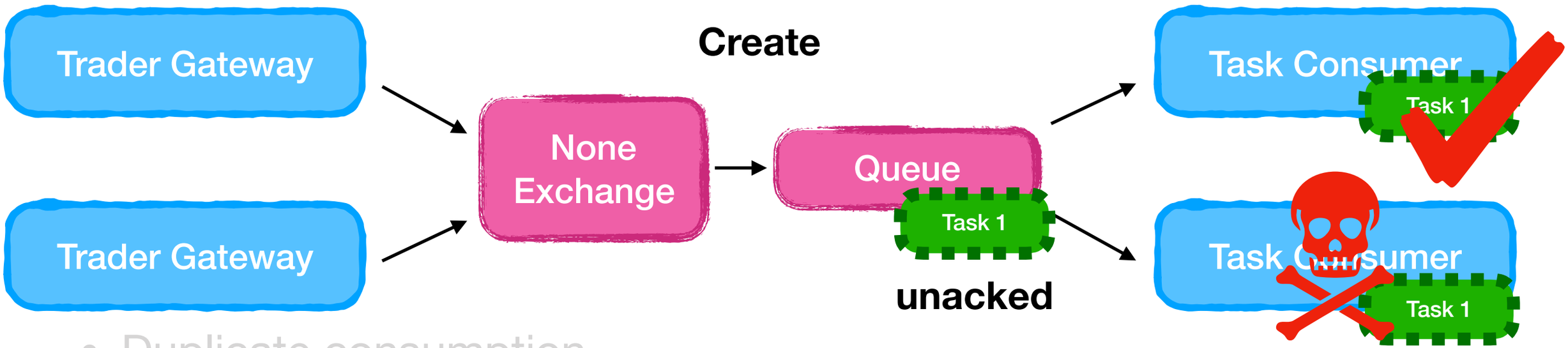
- Duplicate consumption
  - Distributed lock - Redis
  - Check ID
- Message lost
  - AutoAck = false
  - => replica is ok

Situation 2



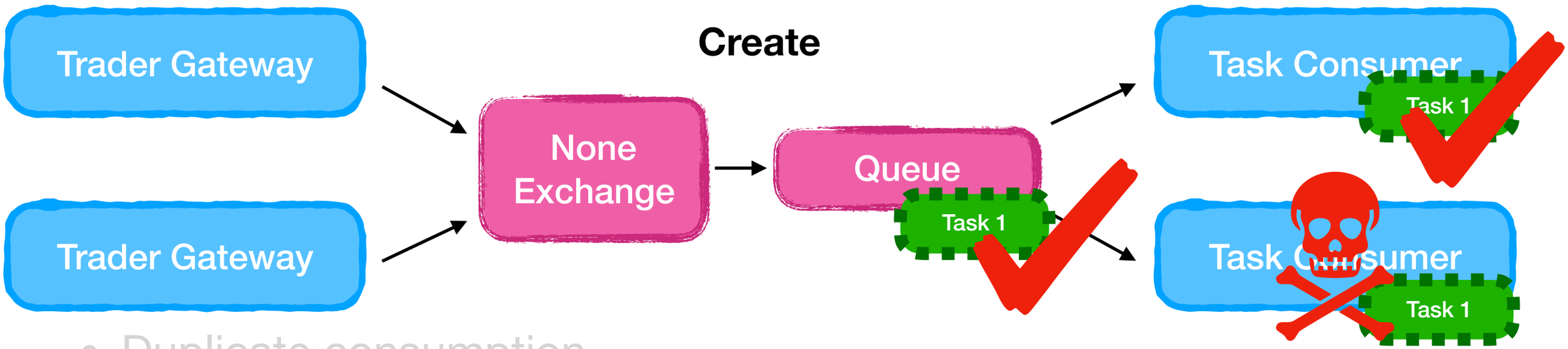
- Duplicate consumption
  - Distributed lock - Redis
  - Check ID
- Message lost
  - AutoAck = false
  - => replica is ok

Situation 2



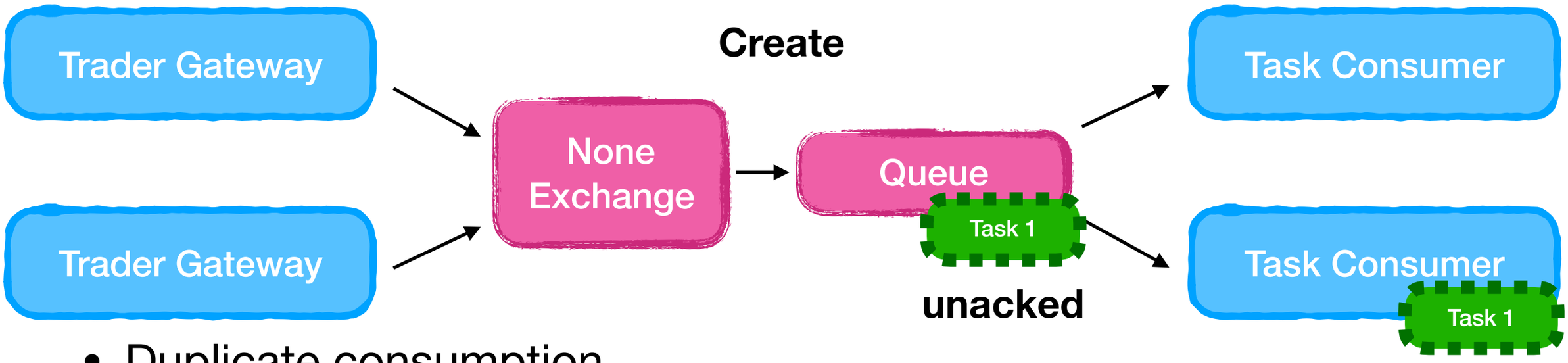
- Duplicate consumption
  - Distributed lock - Redis
  - Check ID
- Message lost
  - AutoAck = false
  - => replica is ok

Situation 2



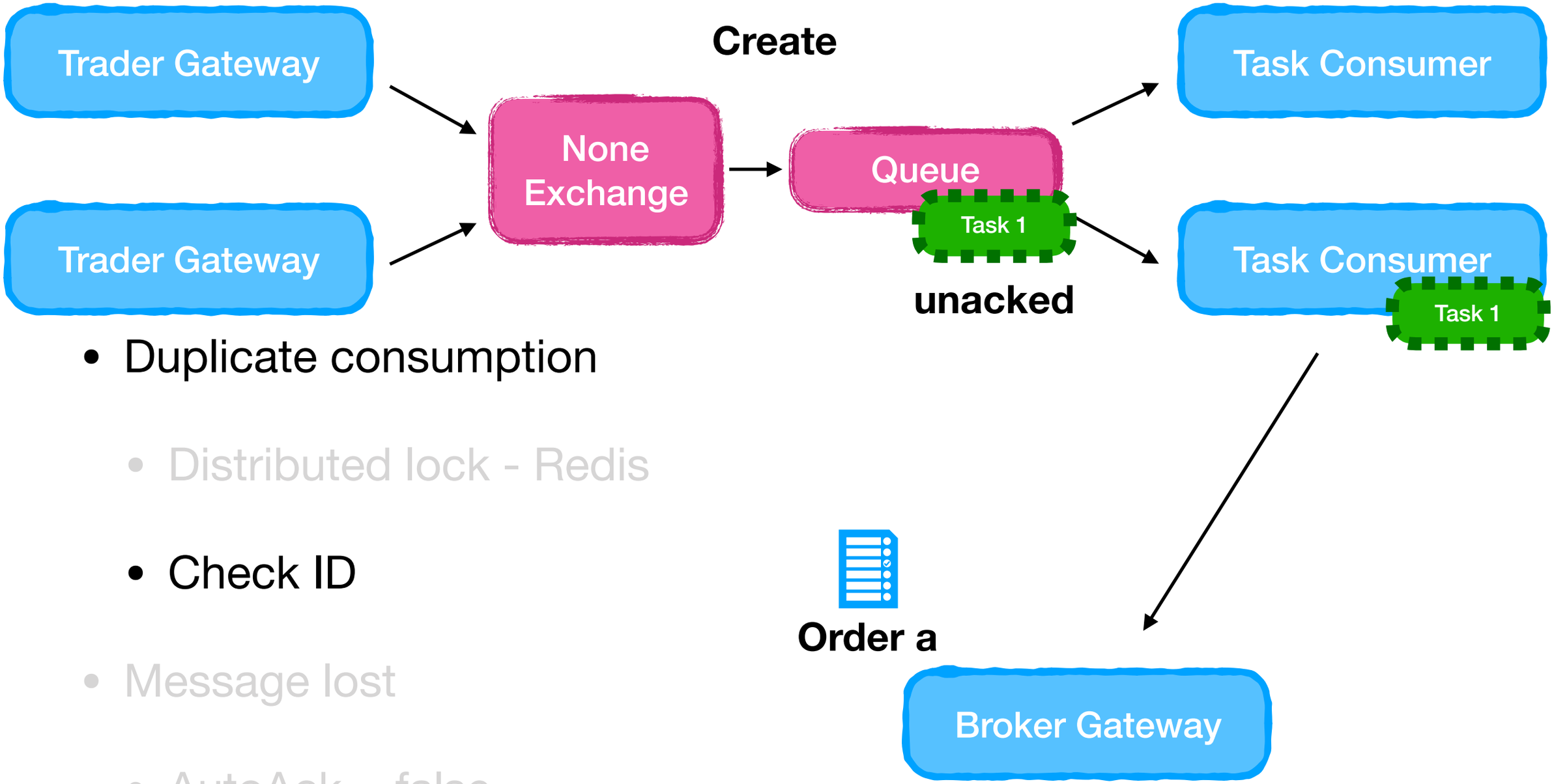
- Duplicate consumption
  - Distributed lock - Redis
  - Check ID
- Message lost
  - AutoAck = false
  - => replica is ok

Situation 3-1



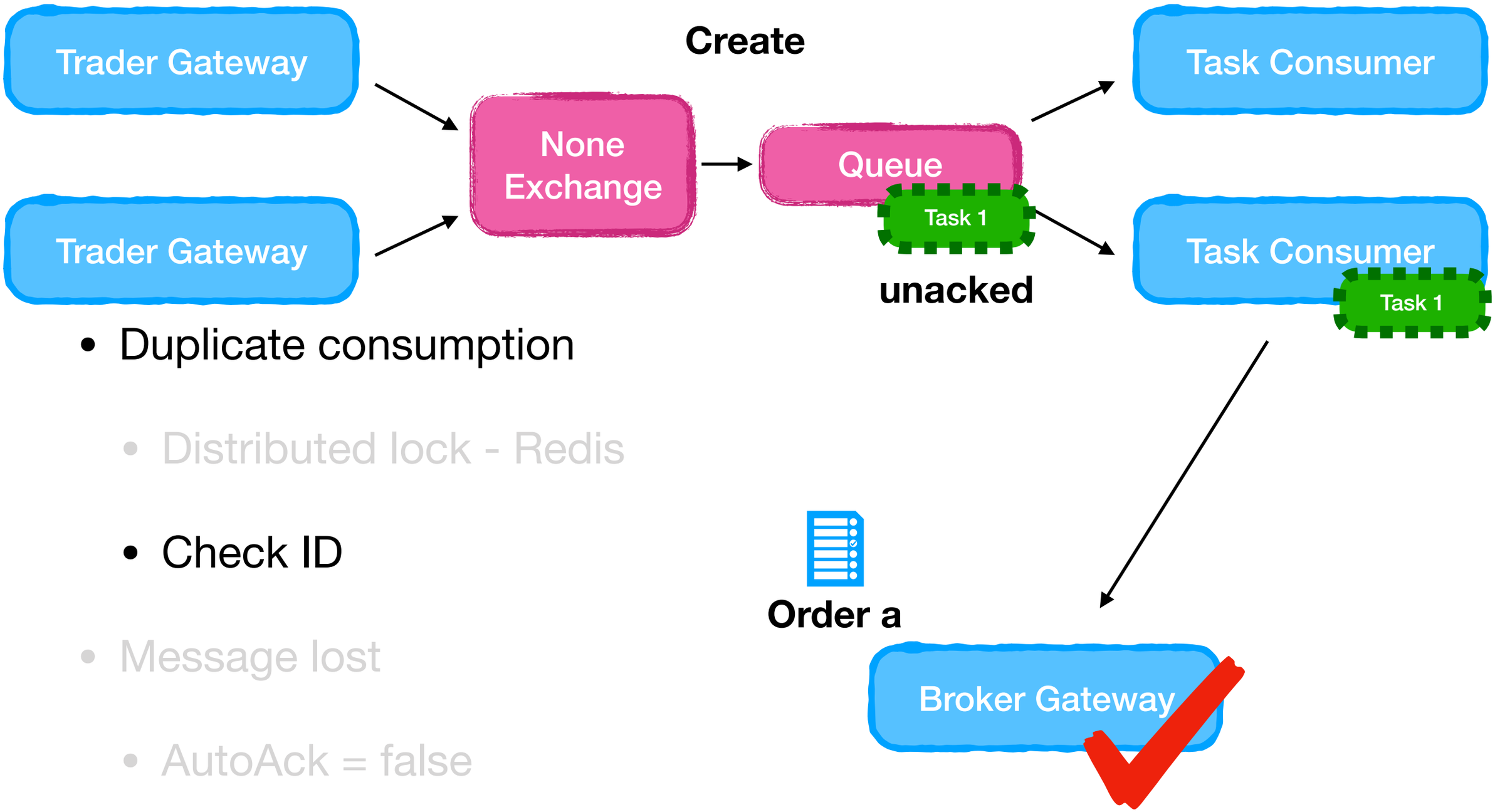
- Duplicate consumption
  - Distributed lock - Redis
  - Check ID
- Message lost
  - AutoAck = false
  - => replica is ok

Situation 3-1



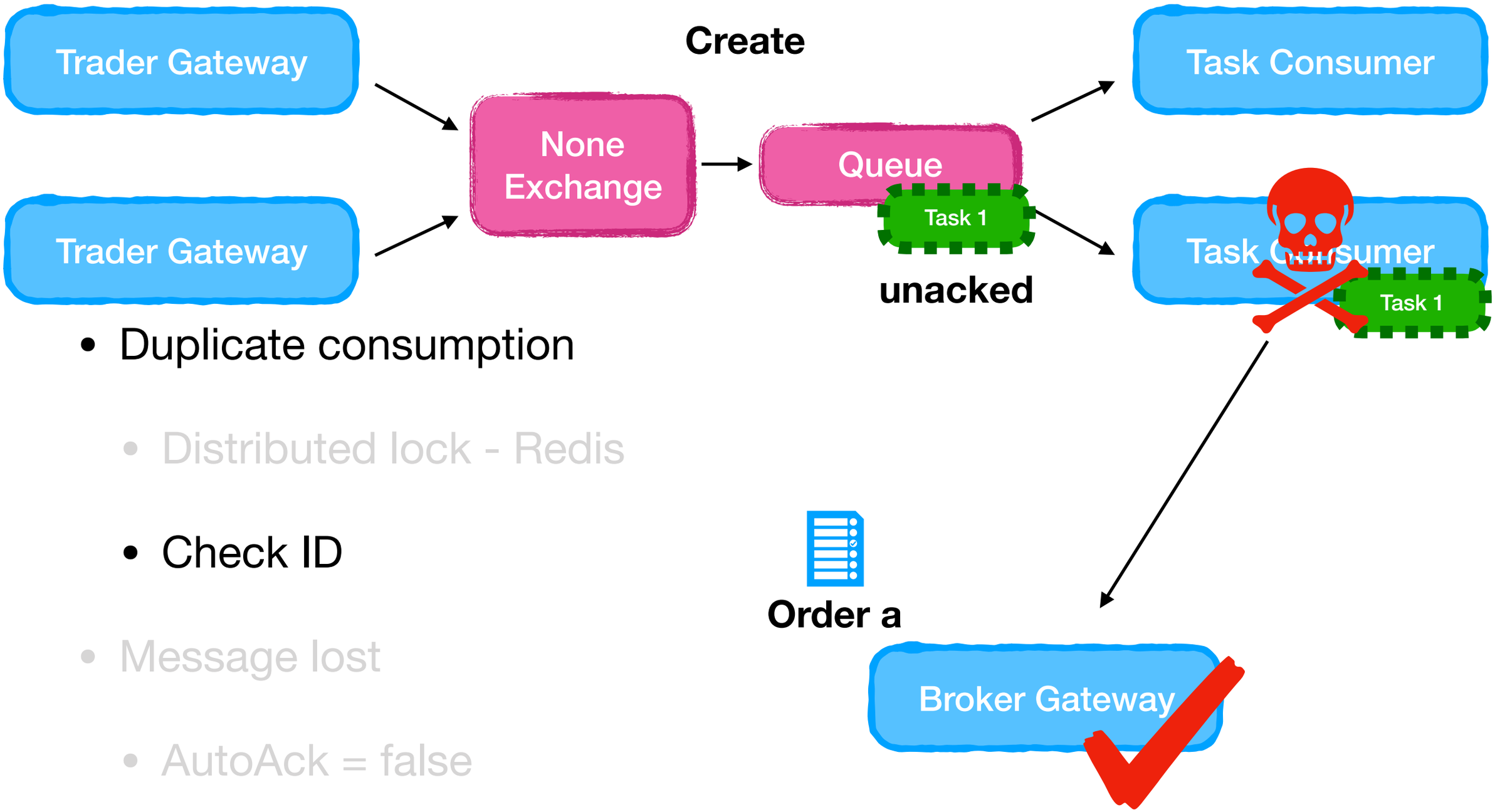
- Duplicate consumption
  - Distributed lock - Redis
  - Check ID
- Message lost
  - AutoAck = false
  - => replica is ok

Situation 3-1



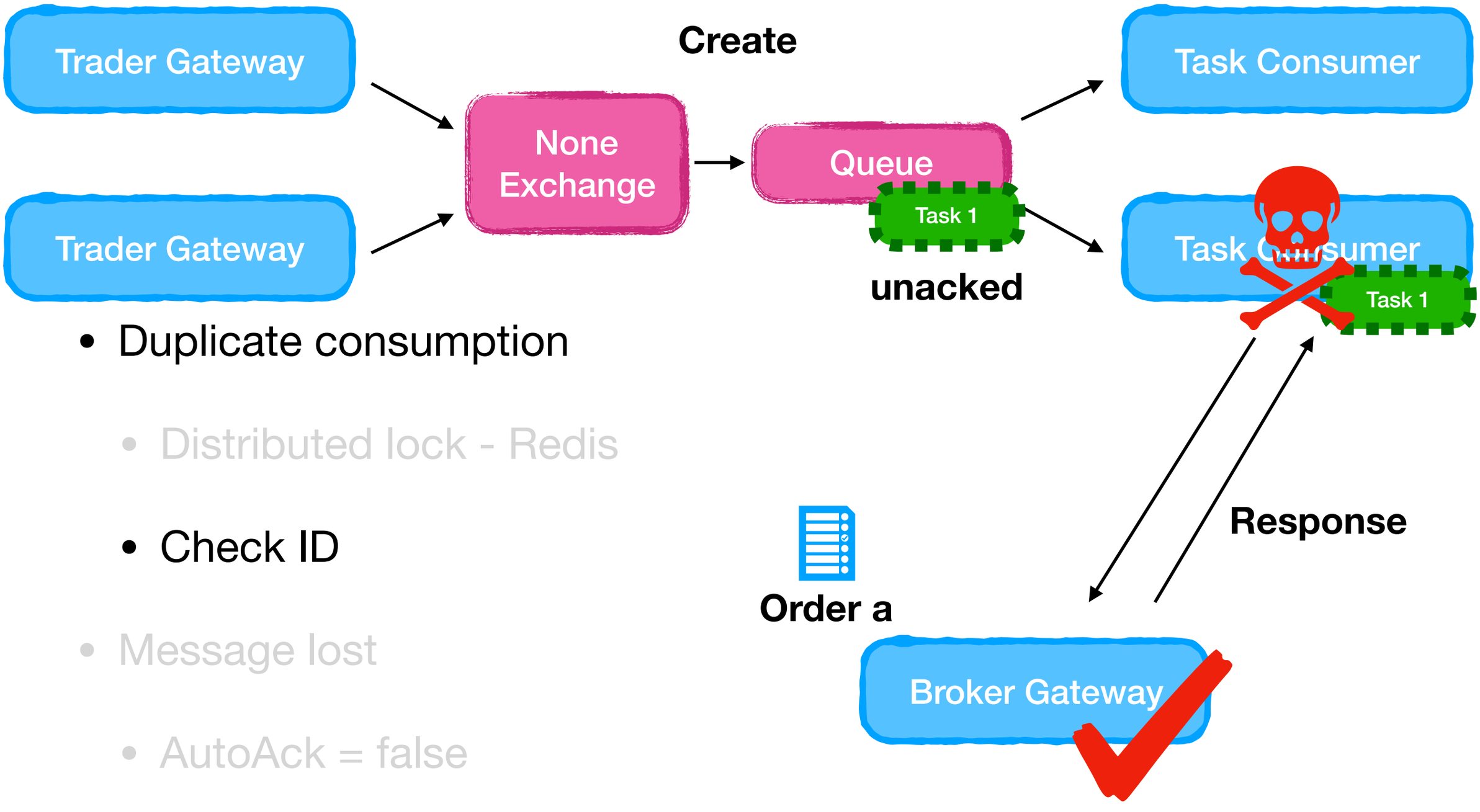
- Duplicate consumption
  - Distributed lock - Redis
  - Check ID
- Message lost
  - AutoAck = false
  - => replica is ok

Situation 3-1



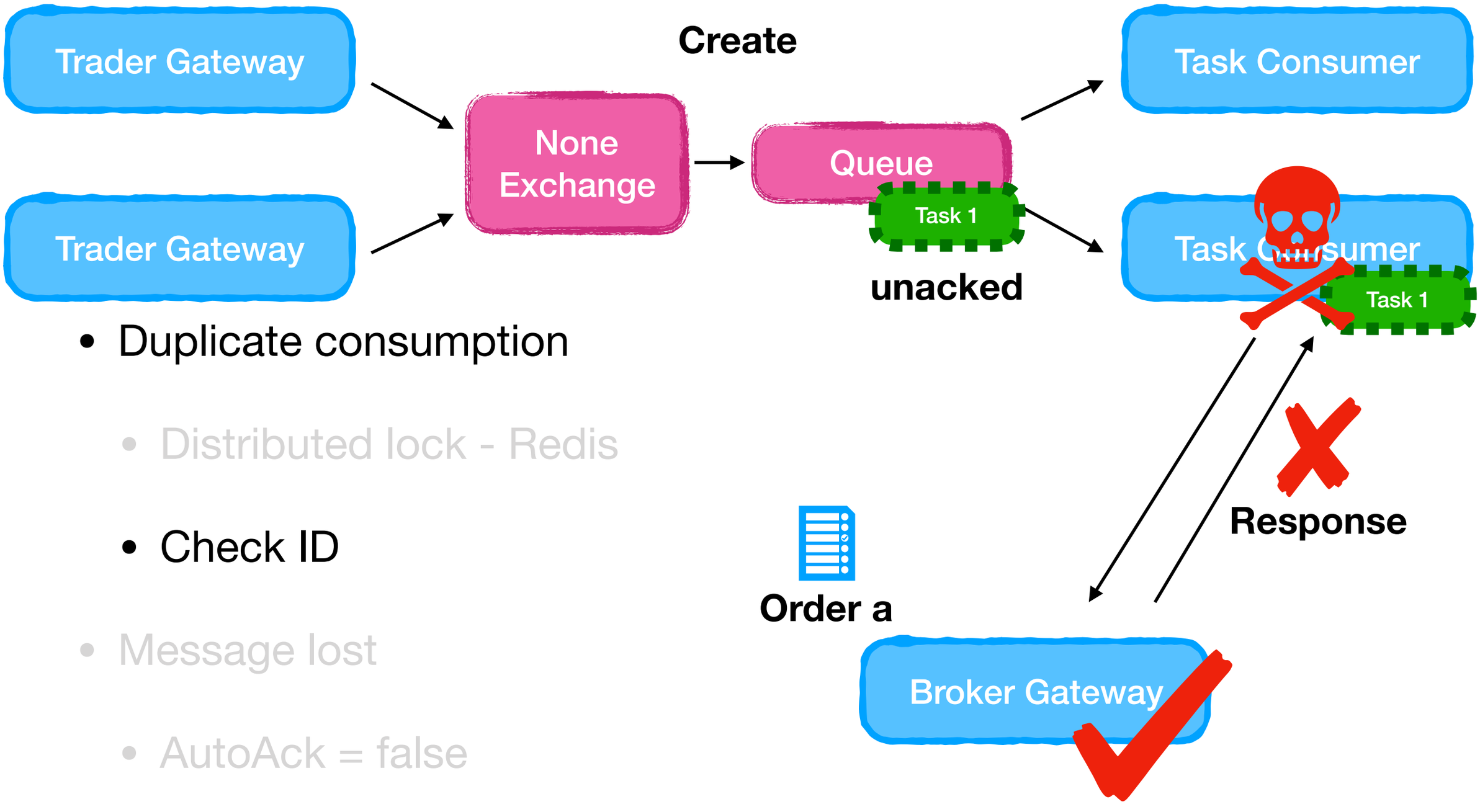
- Duplicate consumption
  - Distributed lock - Redis
  - Check ID
- Message lost
  - AutoAck = false
  - => replica is ok

Situation 3-1



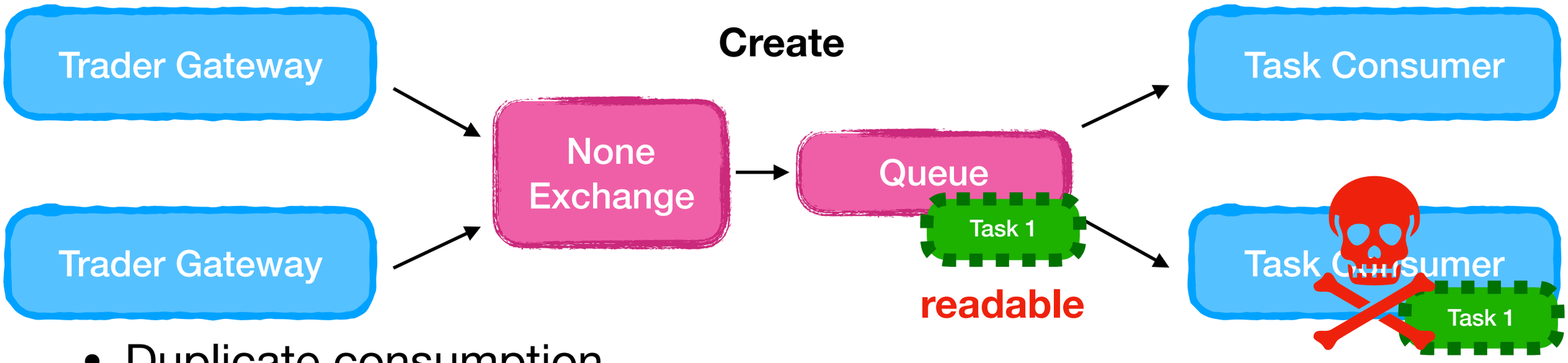
- Duplicate consumption
  - Distributed lock - Redis
  - Check ID
- Message lost
  - AutoAck = false
  - => replica is ok

Situation 3-1



- Duplicate consumption
  - Distributed lock - Redis
  - Check ID
- Message lost
  - AutoAck = false
  - => replica is ok

Situation 3-1

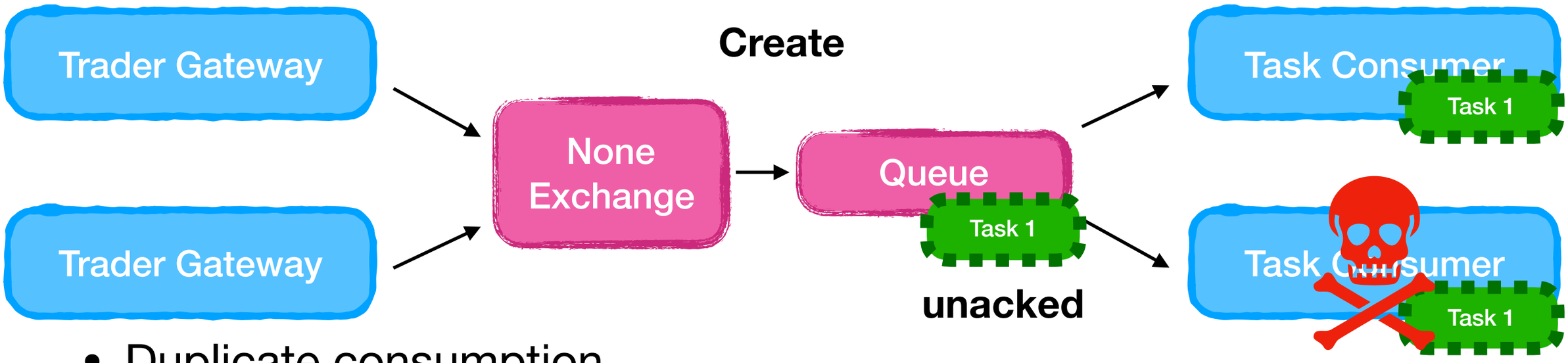


- Duplicate consumption
  - Distributed lock - Redis
  - Check ID
- Message lost
  - AutoAck = false
  - => replica is ok

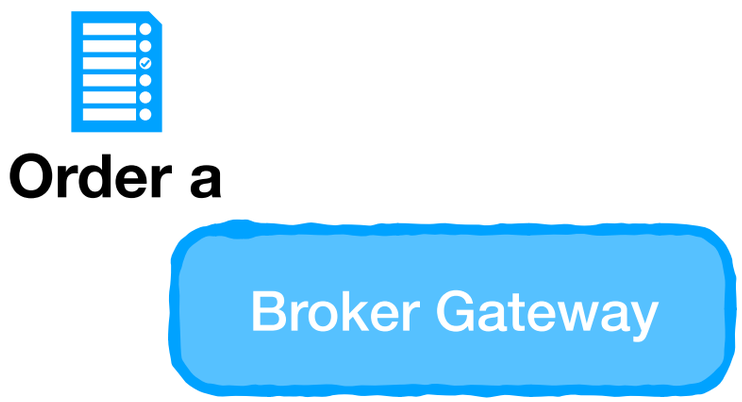
Order a

Broker Gateway

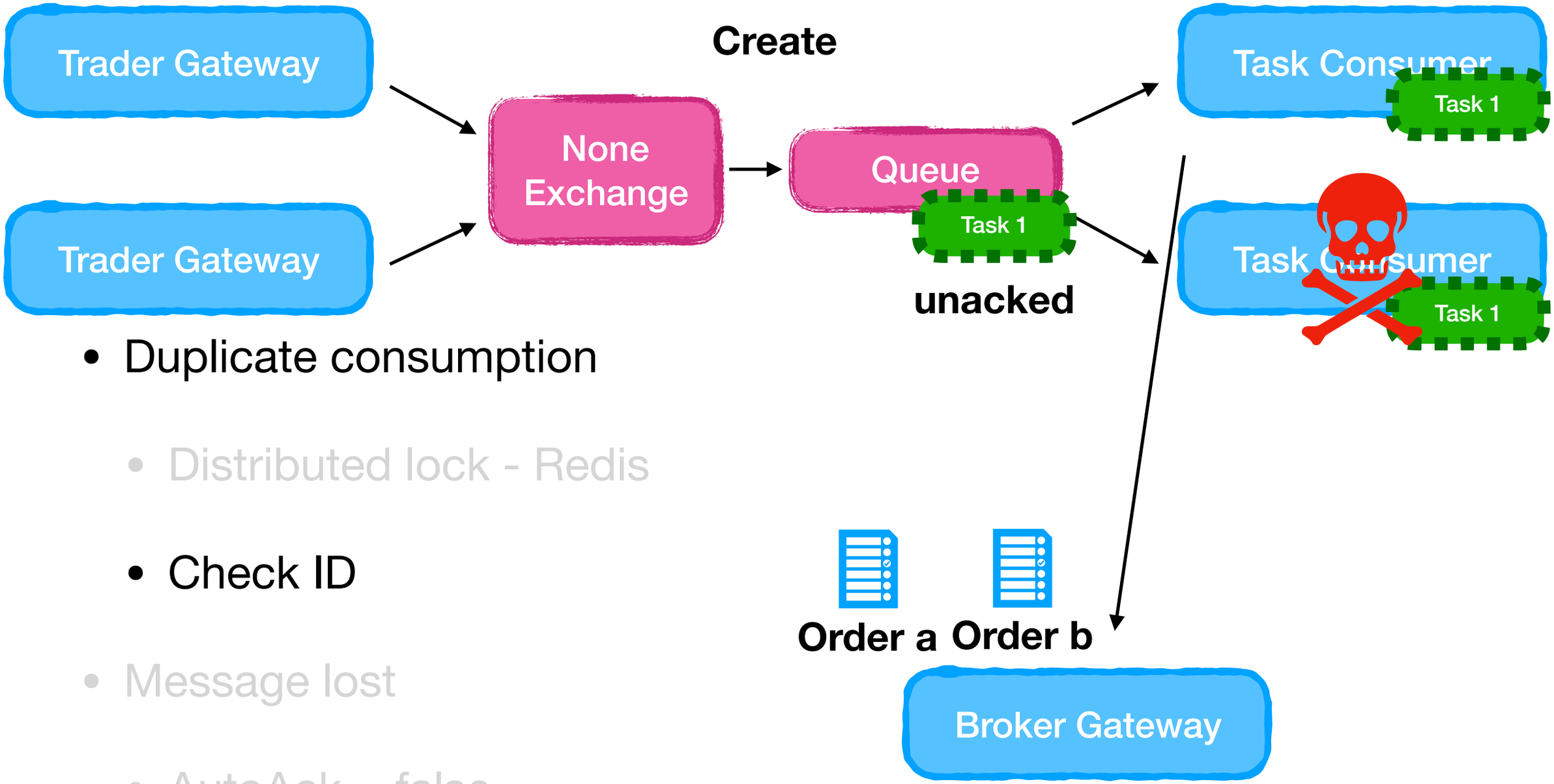
Situation 3-1



- Duplicate consumption
  - Distributed lock - Redis
  - Check ID
- Message lost
  - AutoAck = false
  - => replica is ok

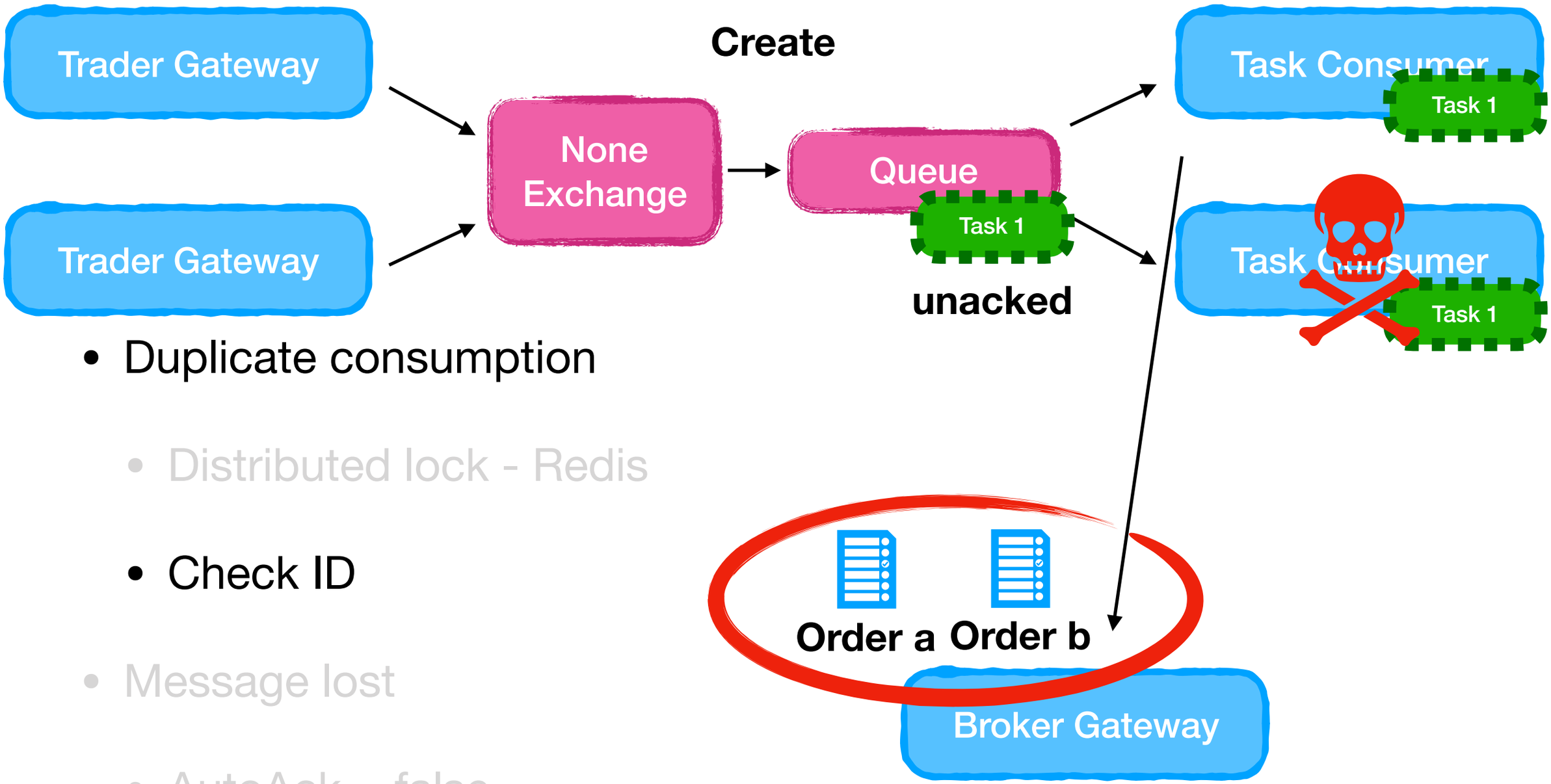


Situation 3-1



- Duplicate consumption
  - Distributed lock - Redis
  - Check ID
- Message lost
  - AutoAck = false
  - => replica is ok

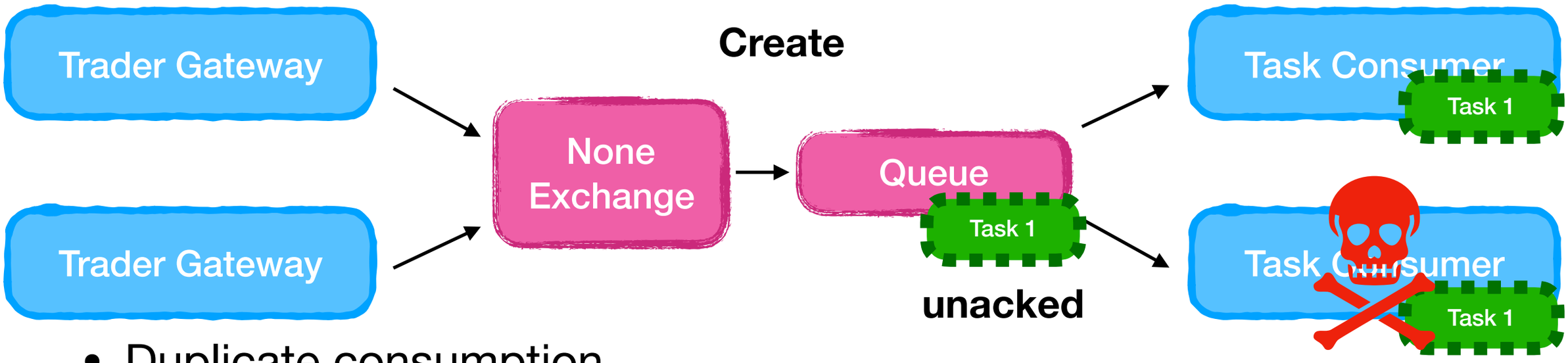
Situation 3-1



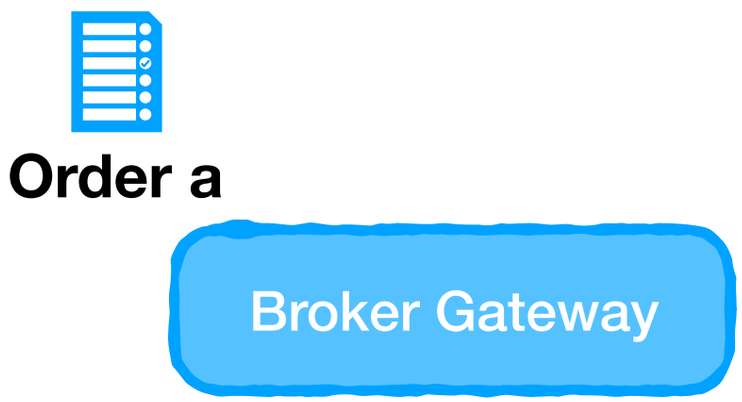
- Duplicate consumption
  - Distributed lock - Redis
  - Check ID
- Message lost
  - AutoAck = false
  - => replica is ok

Task 1 is consumed twice !!!

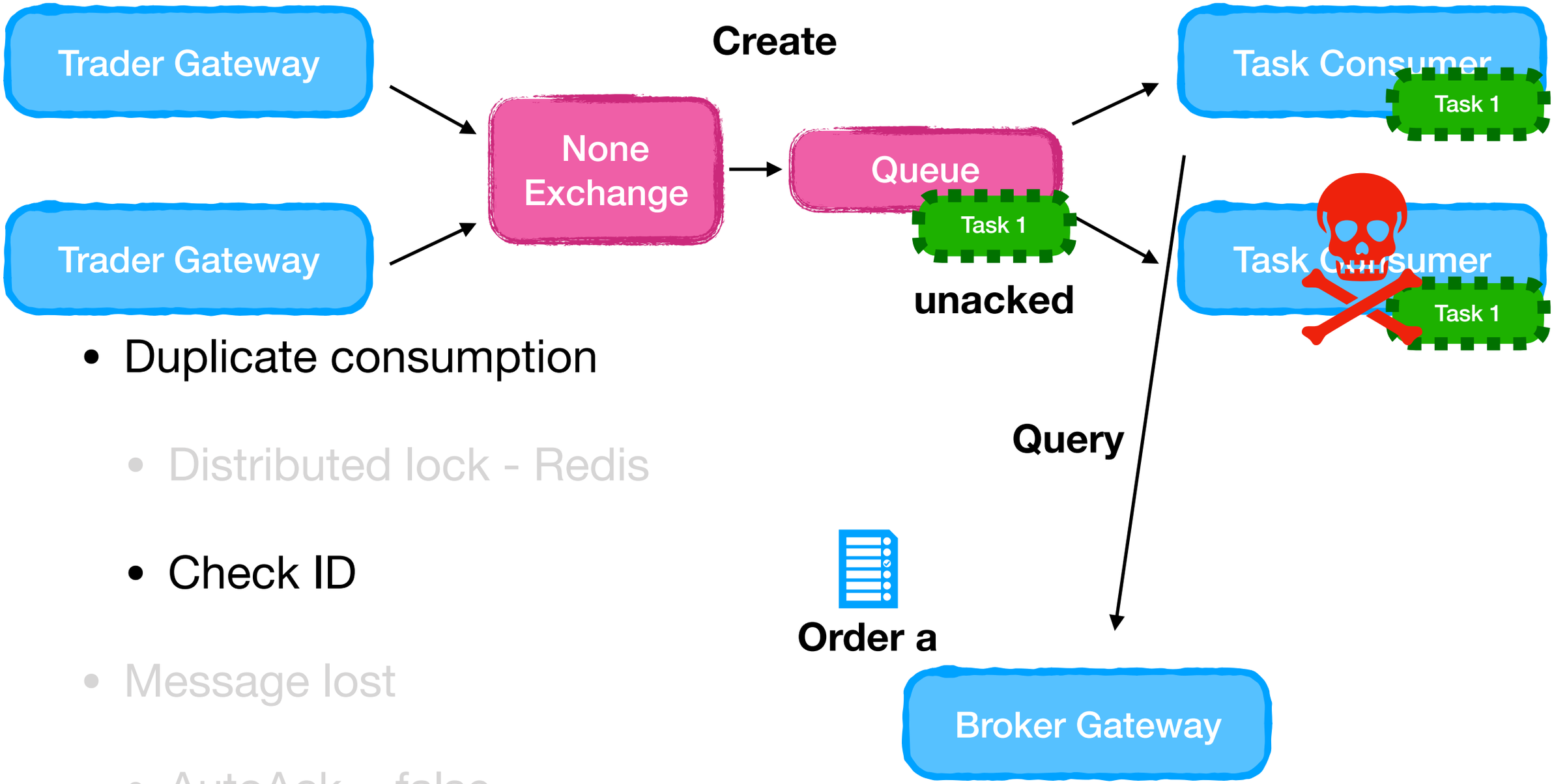
Situation 3-2



- Duplicate consumption
  - Distributed lock - Redis
  - Check ID
- Message lost
  - AutoAck = false
  - => replica is ok

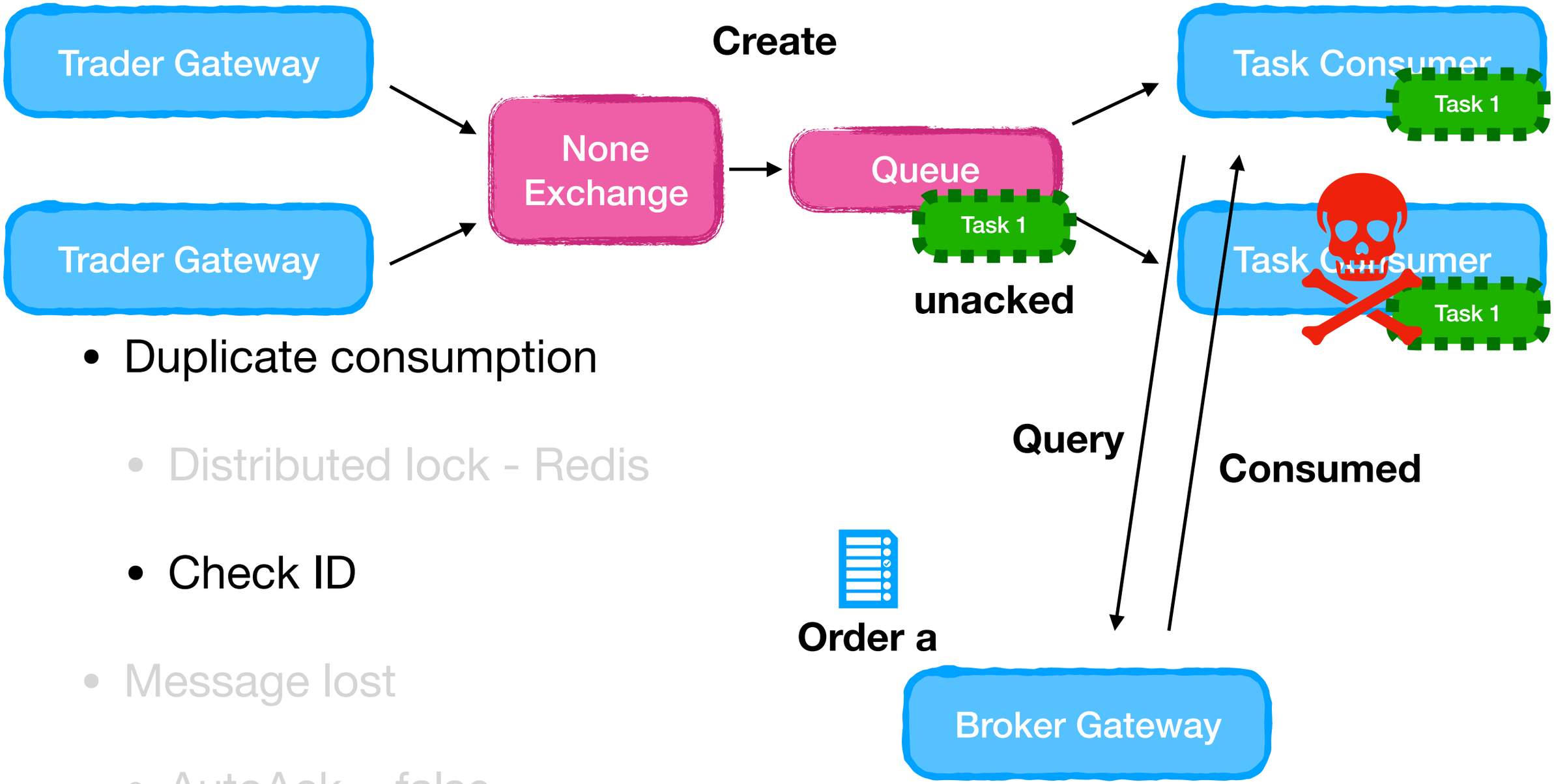


Situation 3-2



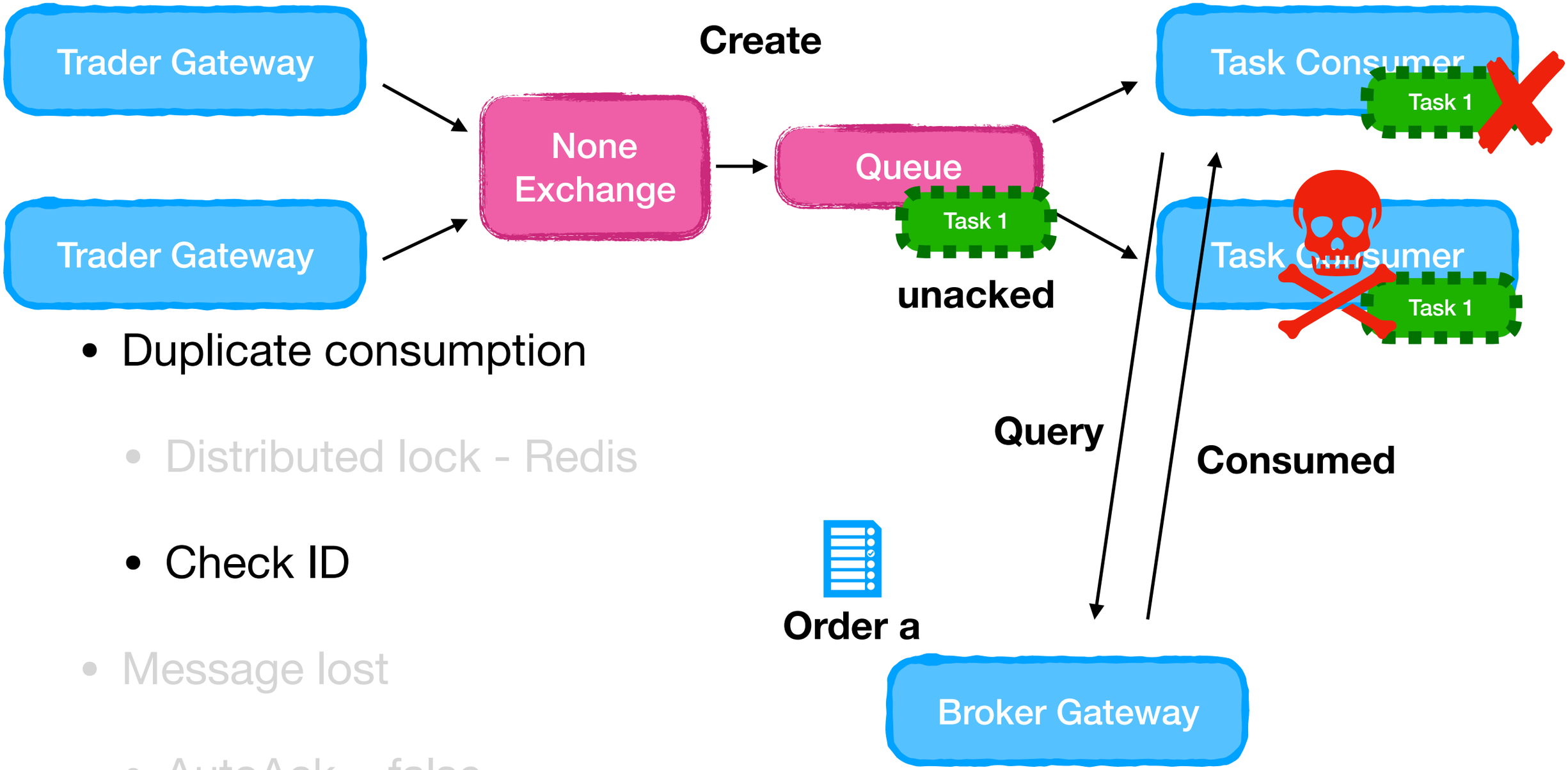
- Duplicate consumption
  - Distributed lock - Redis
  - Check ID
- Message lost
  - AutoAck = false
  - => replica is ok

Situation 3-2



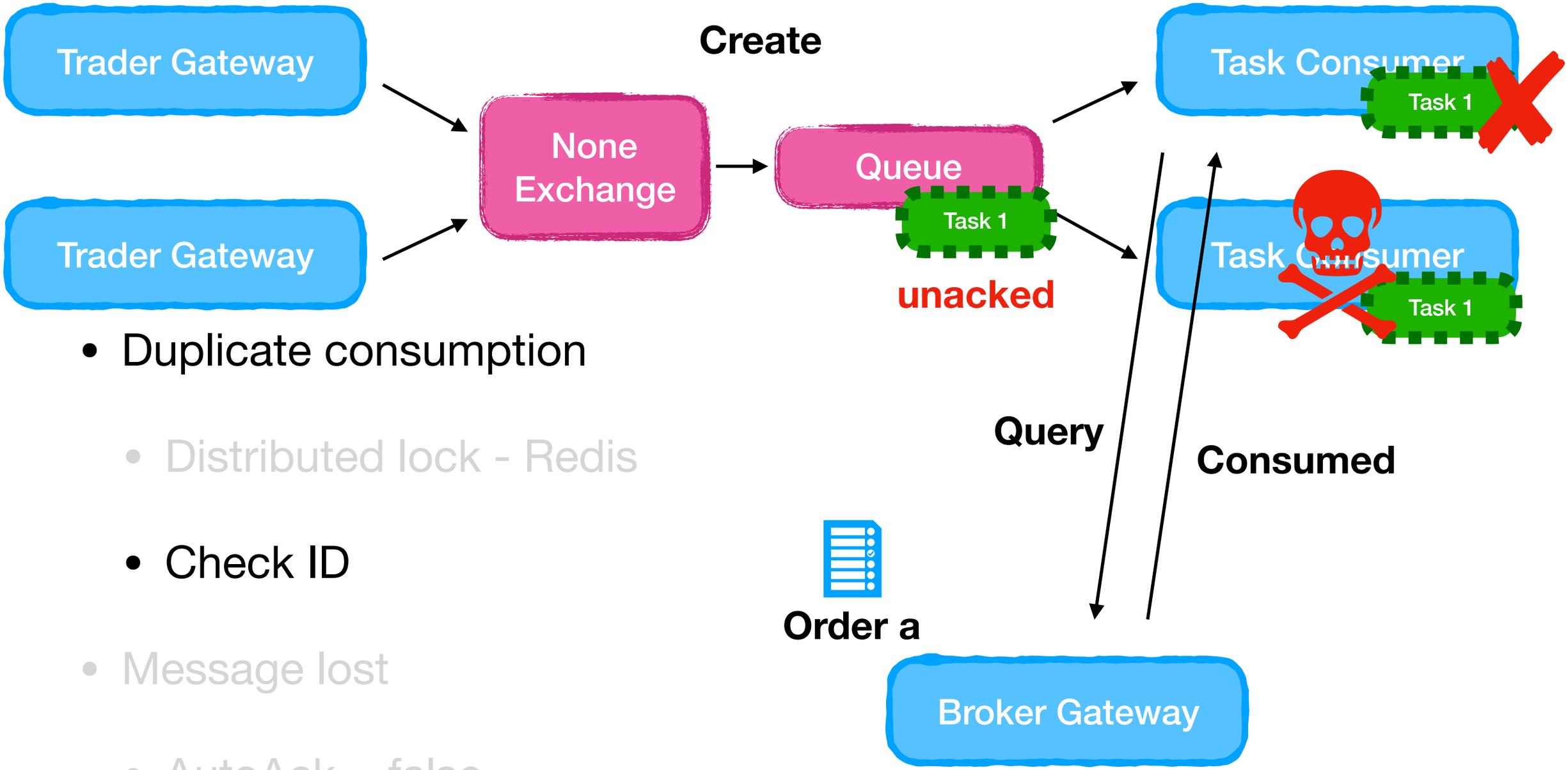
- Duplicate consumption
  - Distributed lock - Redis
  - Check ID
- Message lost
  - AutoAck = false
  - => replica is ok

Situation 3-2



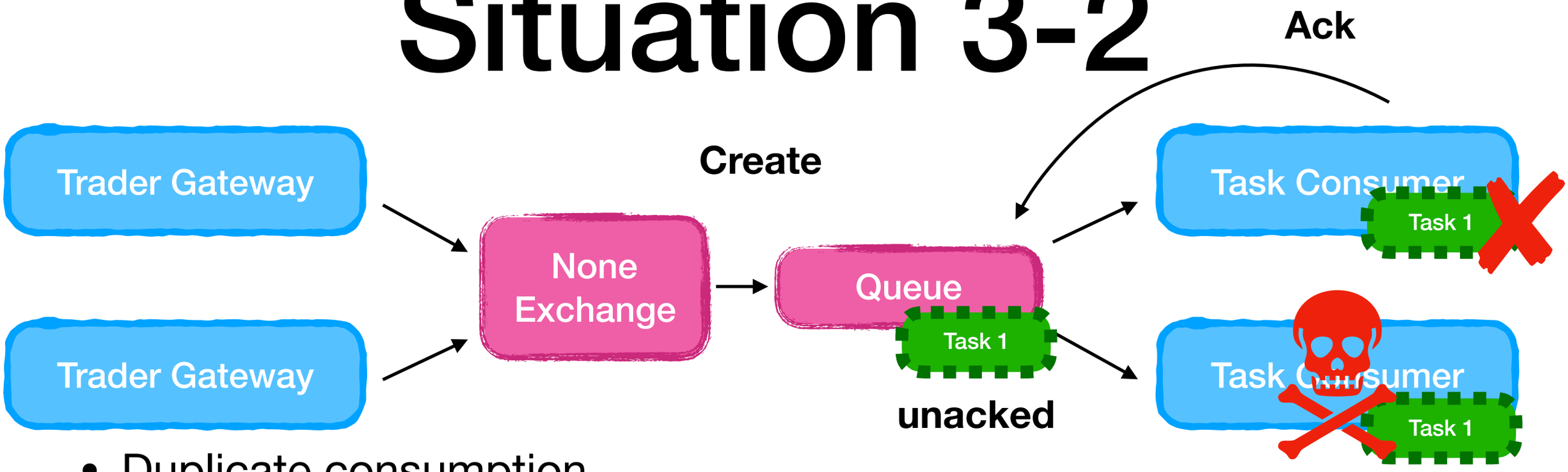
- Duplicate consumption
  - Distributed lock - Redis
  - Check ID
- Message lost
  - AutoAck = false
  - => replica is ok

Situation 3-2

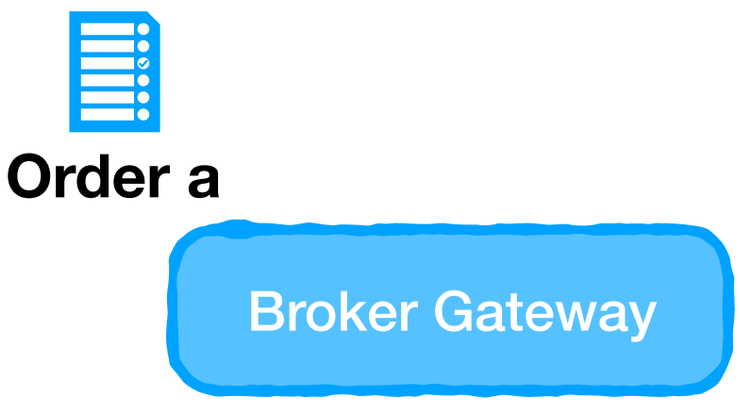


- Duplicate consumption
  - Distributed lock - Redis
  - Check ID
- Message lost
  - AutoAck = false
  - => replica is ok

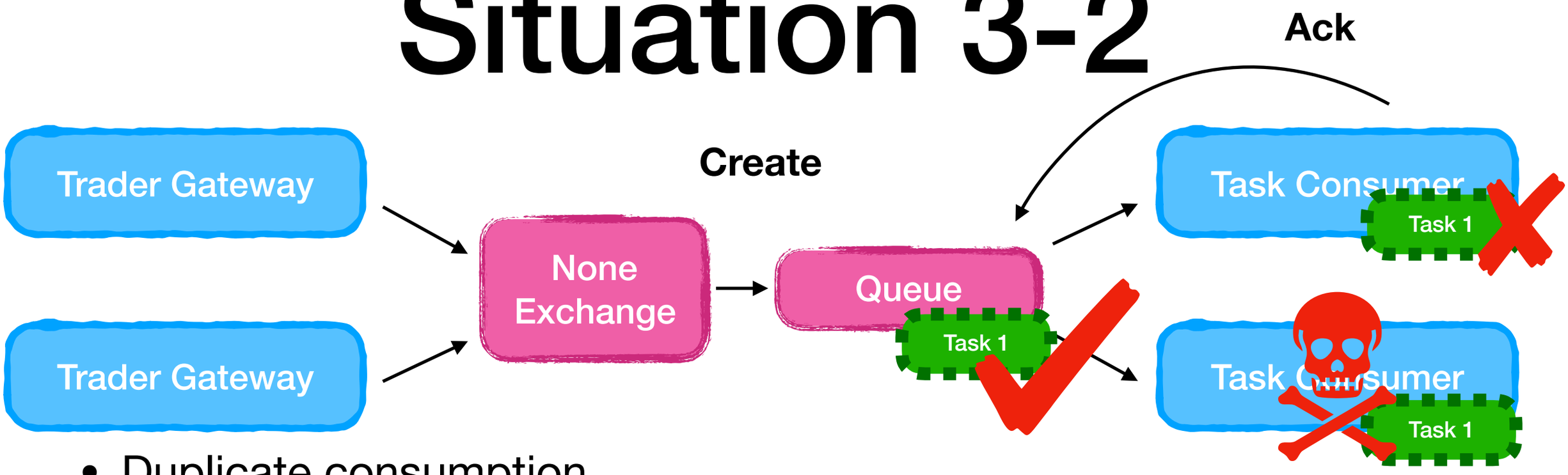
Situation 3-2



- Duplicate consumption
  - Distributed lock - Redis
  - Check ID
- Message lost
  - AutoAck = false
  - => replica is ok



Situation 3-2

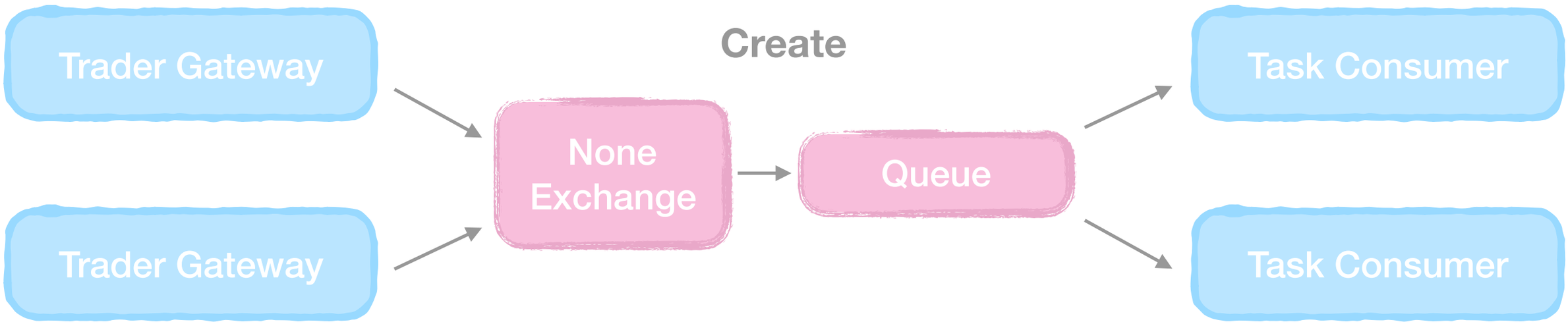


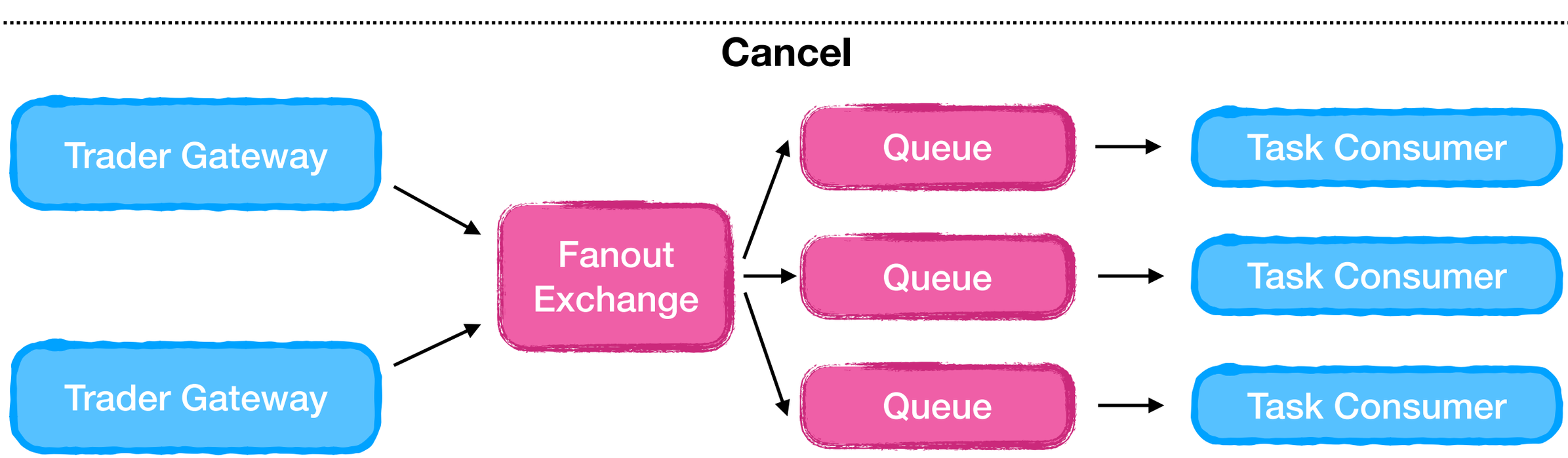
- Duplicate consumption
  - Distributed lock - Redis
  - Check ID
- Message lost
  - AutoAck = false
  - => replica is ok

Order a

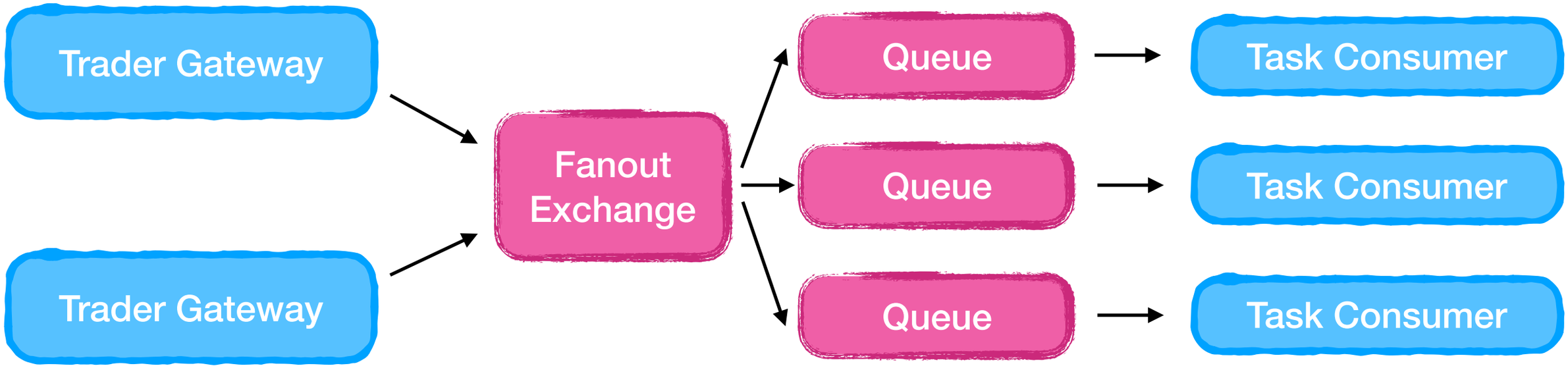
Broker Gateway

Task Consumer



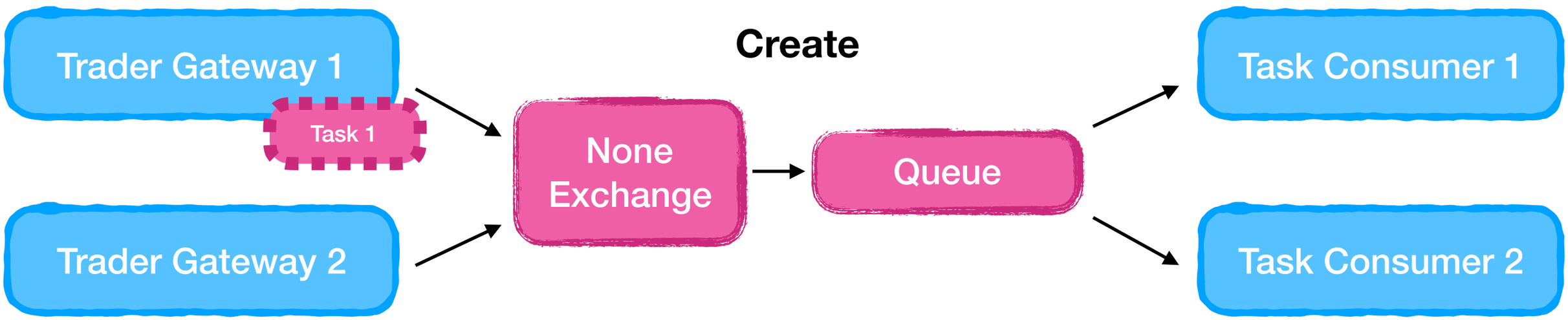


Task Consumer

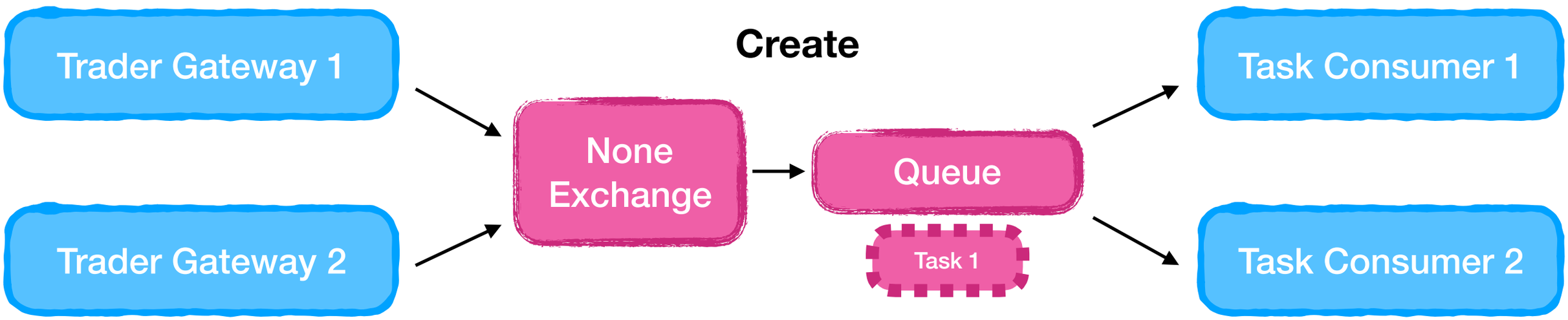


- Why fanout exchange

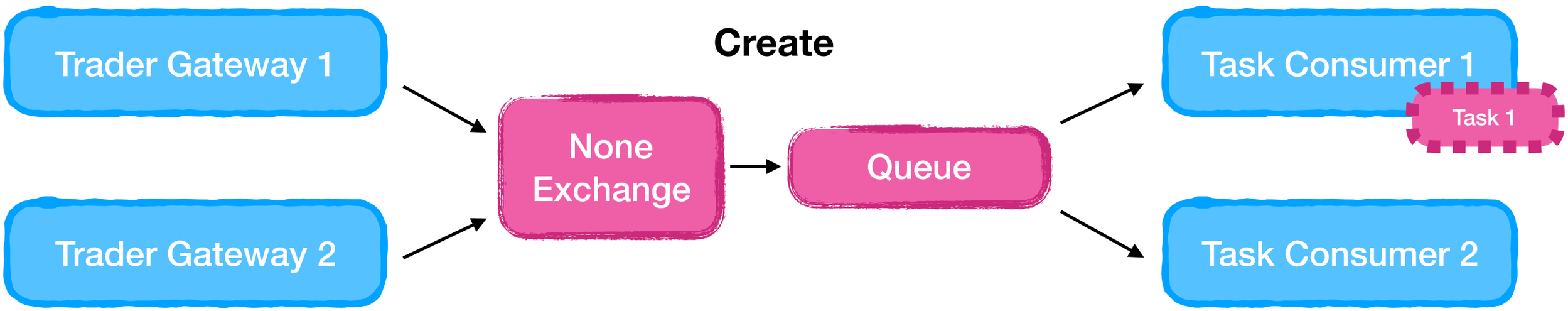
Situation 1 - Without Fanout



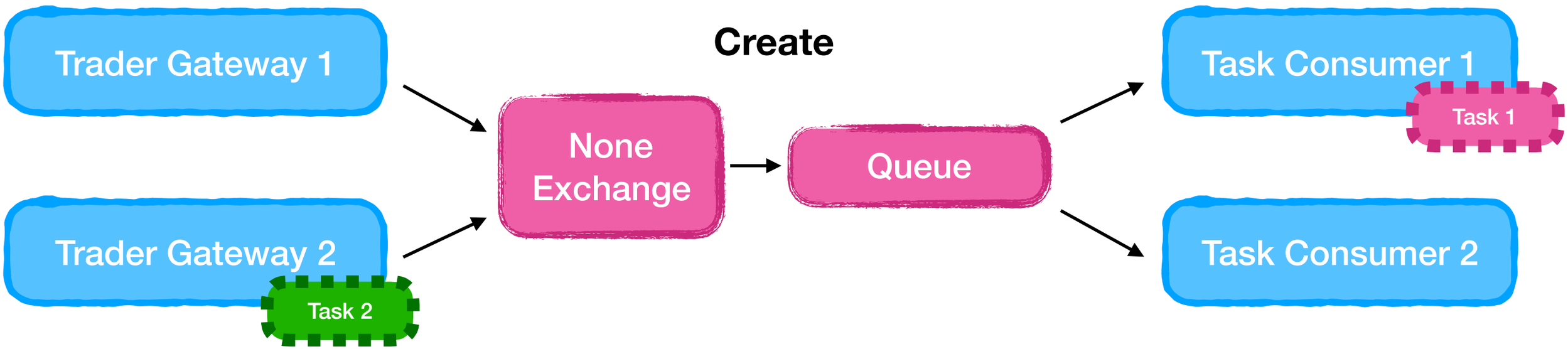
Situation 1 - Without Fanout



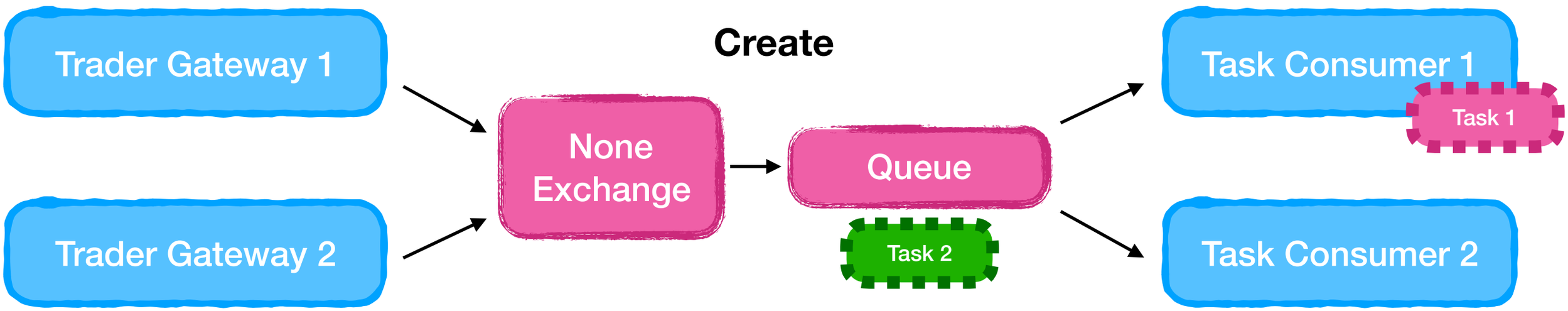
Situation 1 - Without Fanout



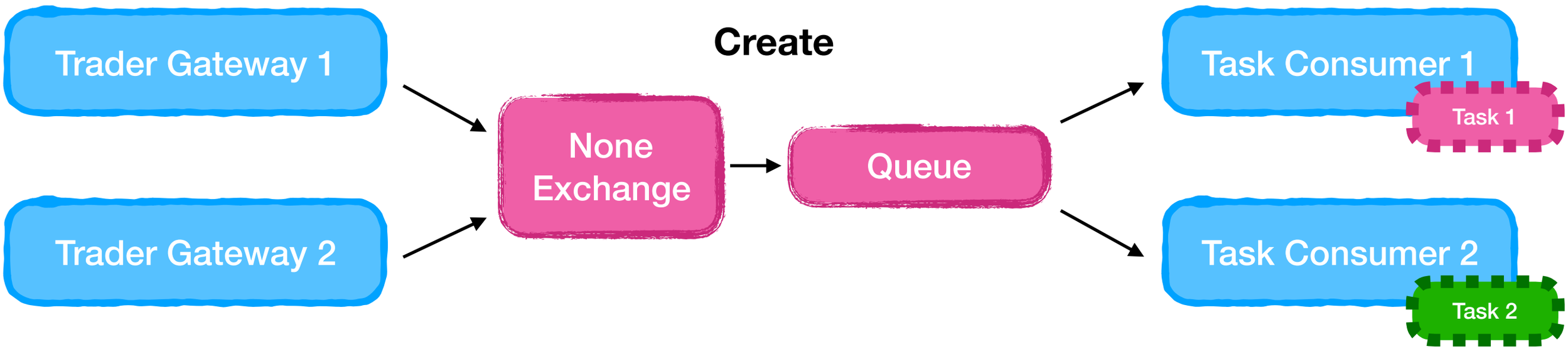
Situation 1 - Without Fanout



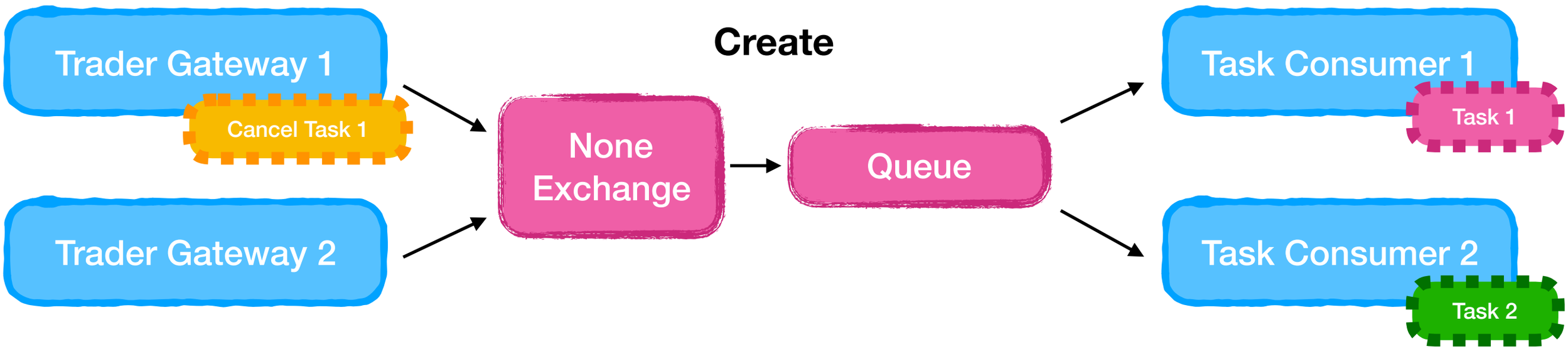
Situation 1 - Without Fanout



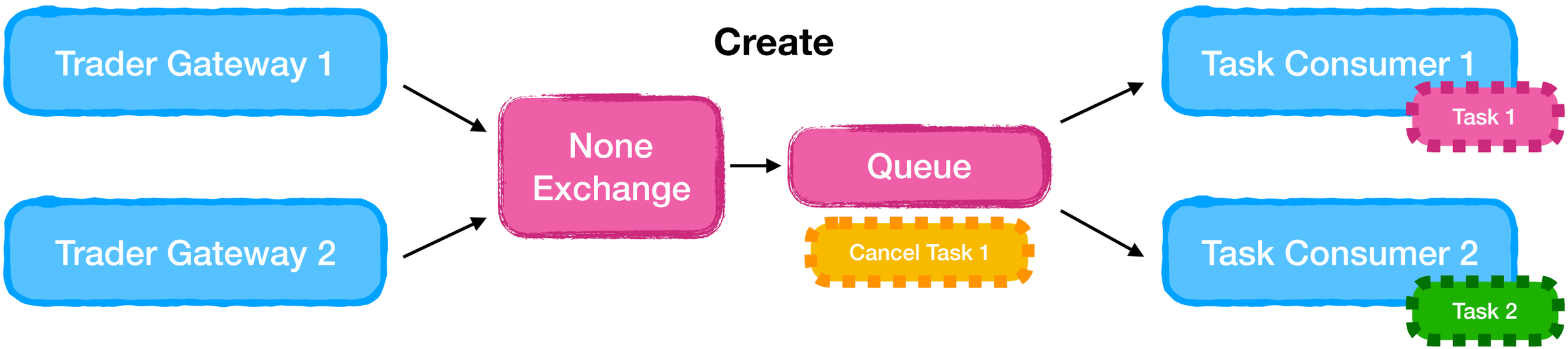
Situation 1 - Without Fanout



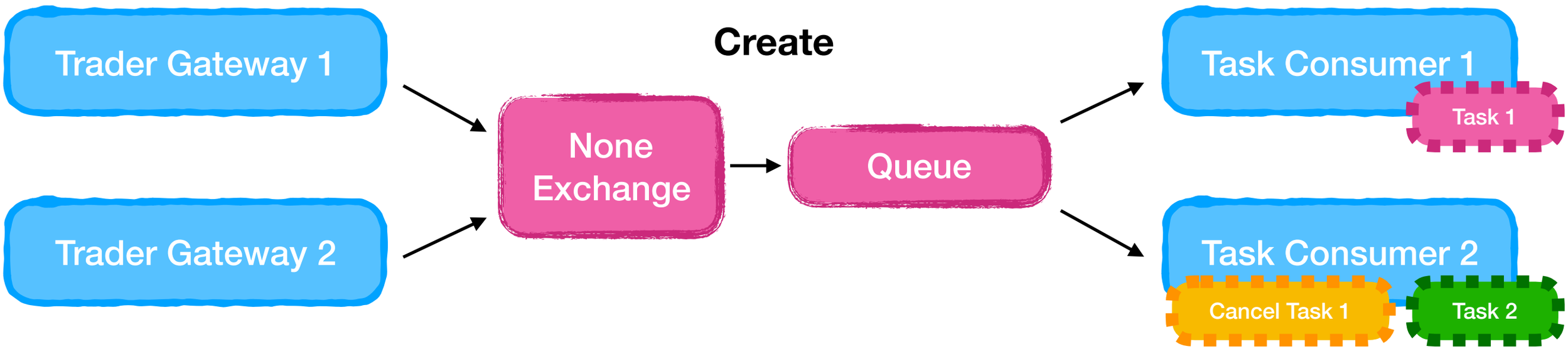
Situation 1 - Without Fanout



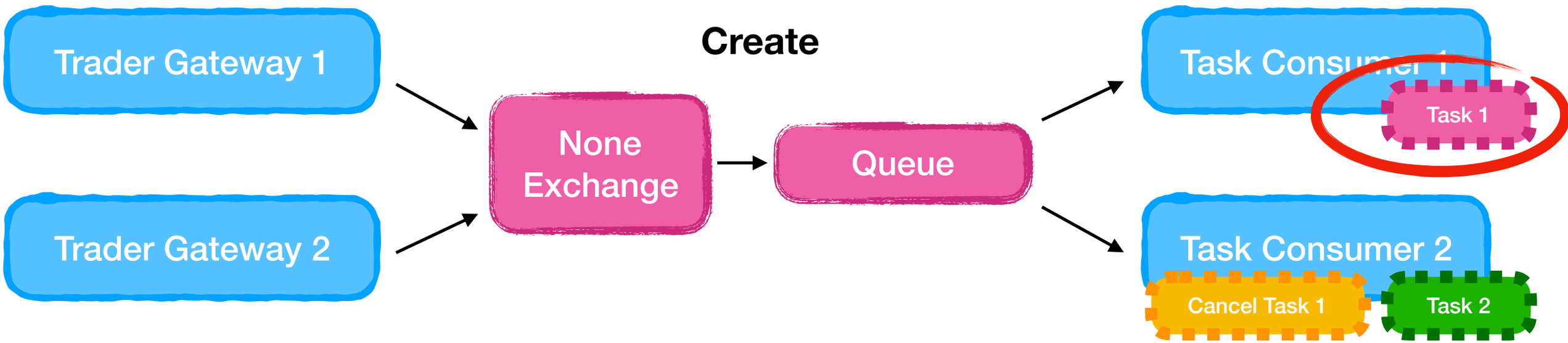
Situation 1 - Without Fanout



Situation 1 - Without Fanout

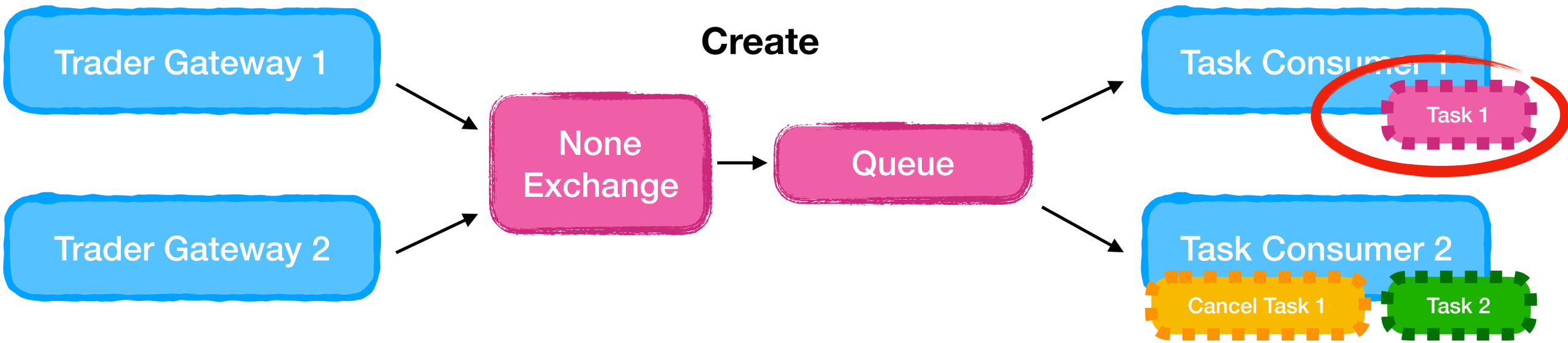


Situation 1 - Without Fanout



But Task 1 isn’t cancelled !!!

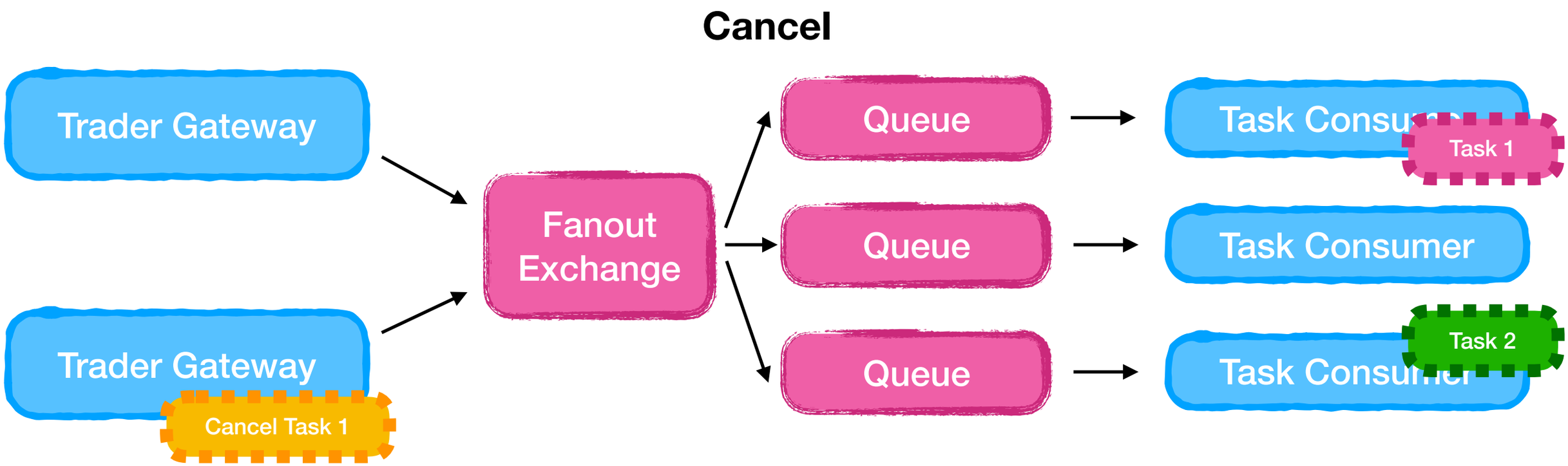
Situation 1 - Without Fanout



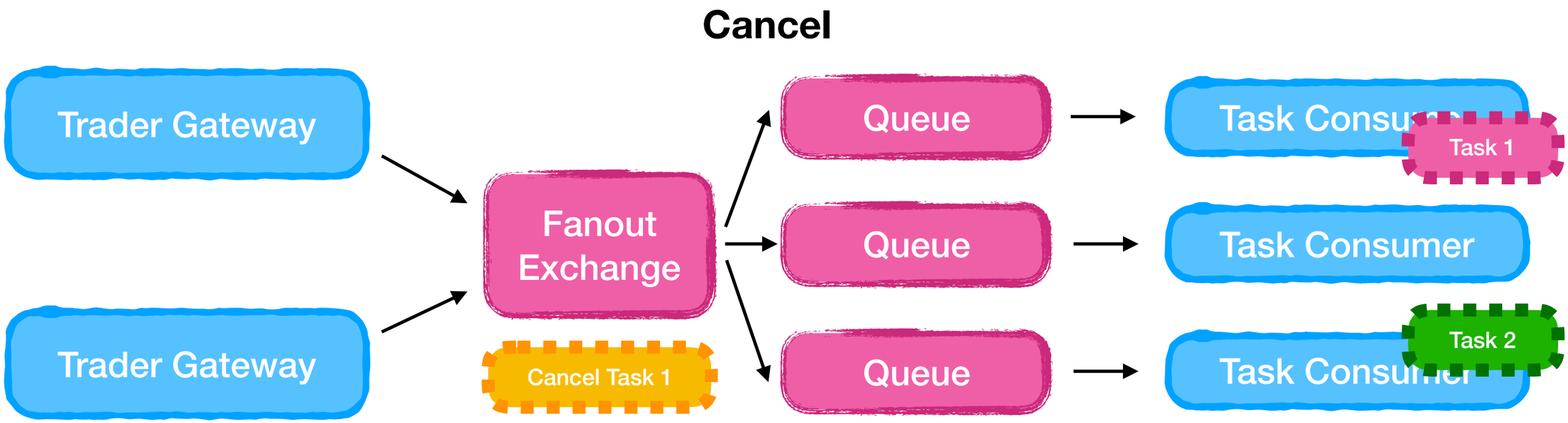
But Task 1 isn’t cancelled !!!

We don’t know where the task is

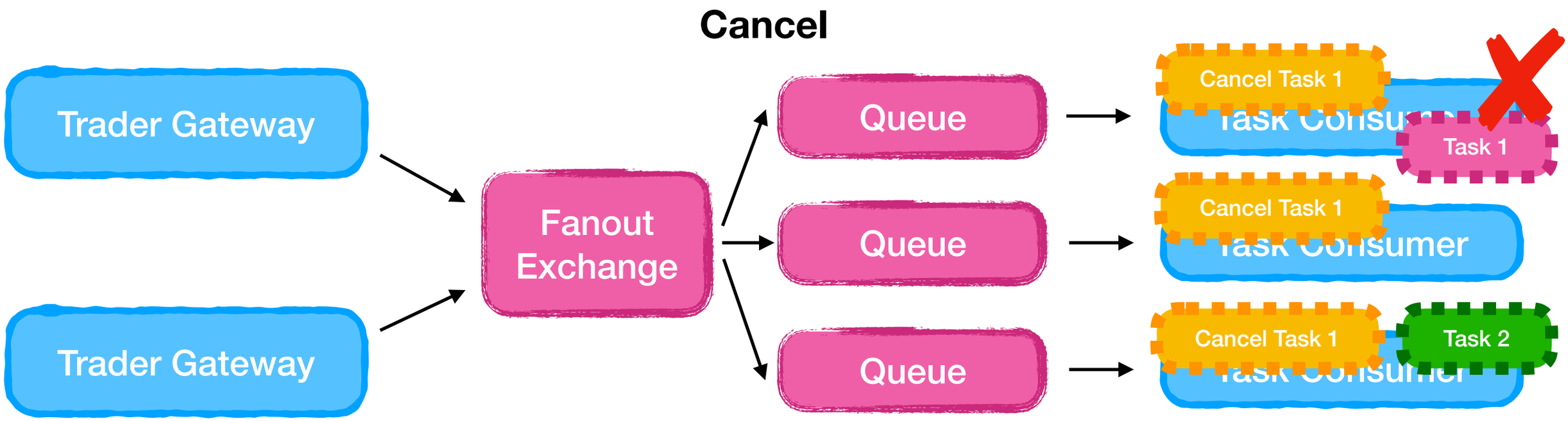
Situation 2 - fanout exchange



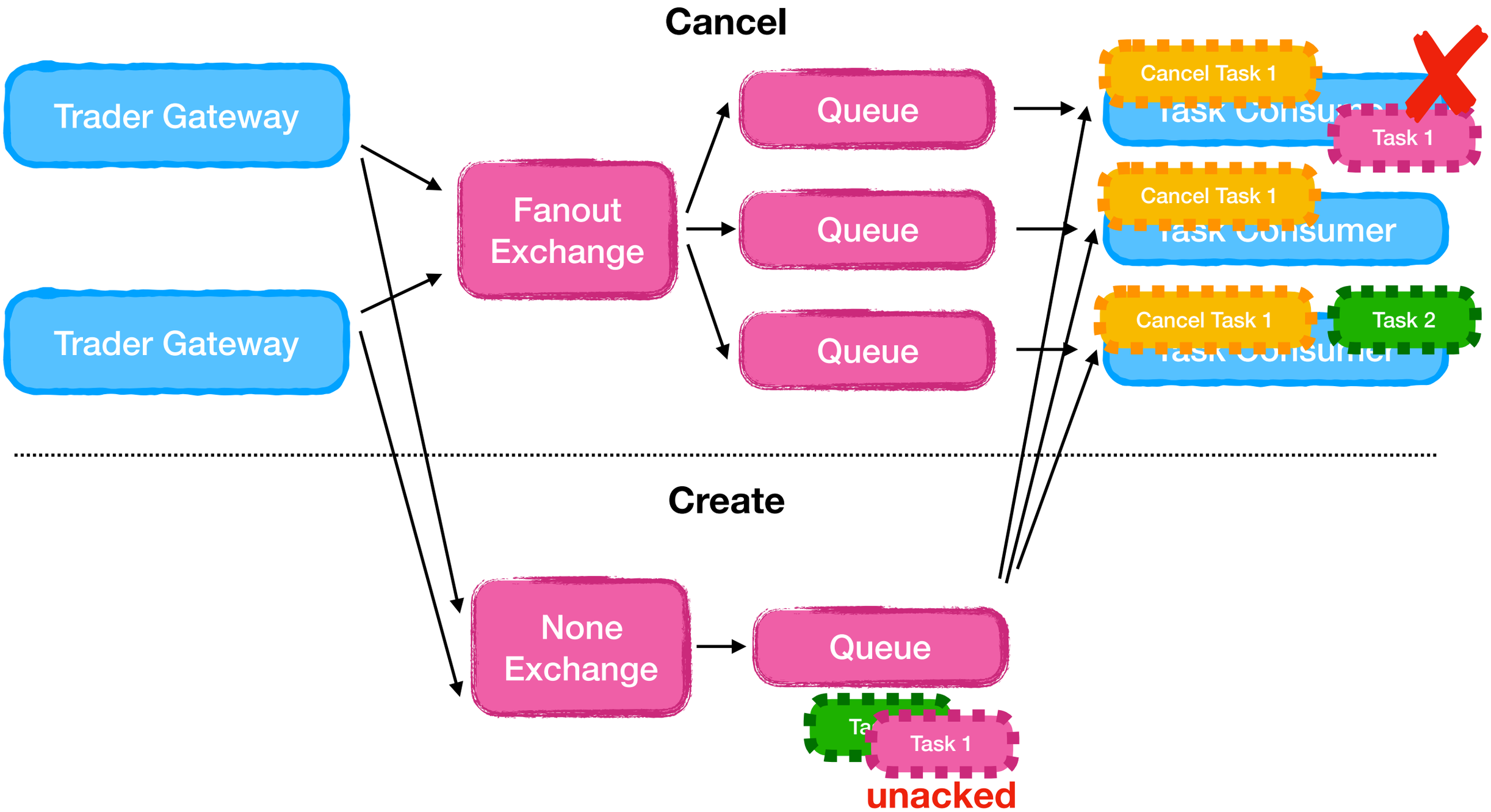
Situation 2 - fanout exchange



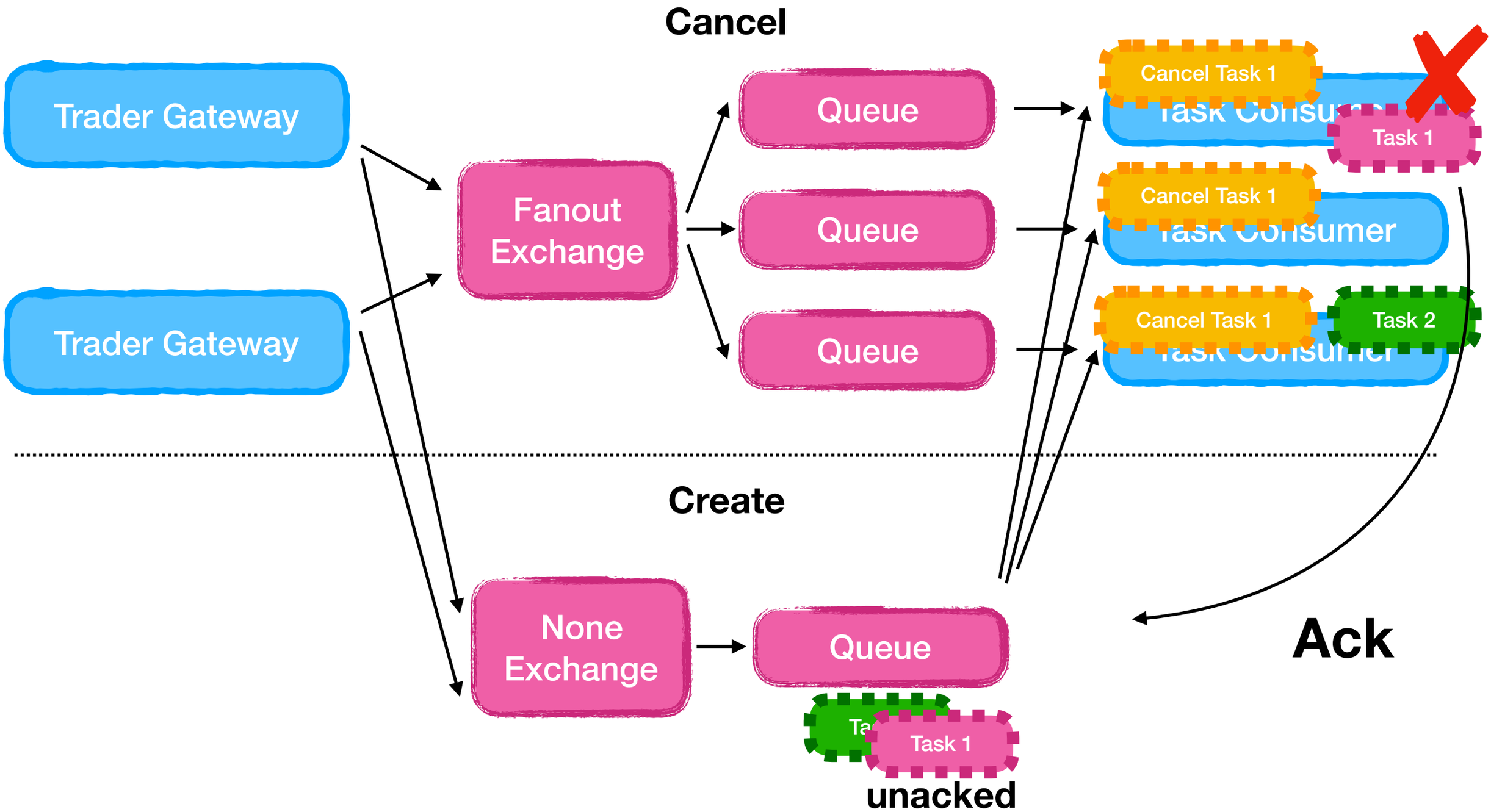
Situation 2 - fanout exchange



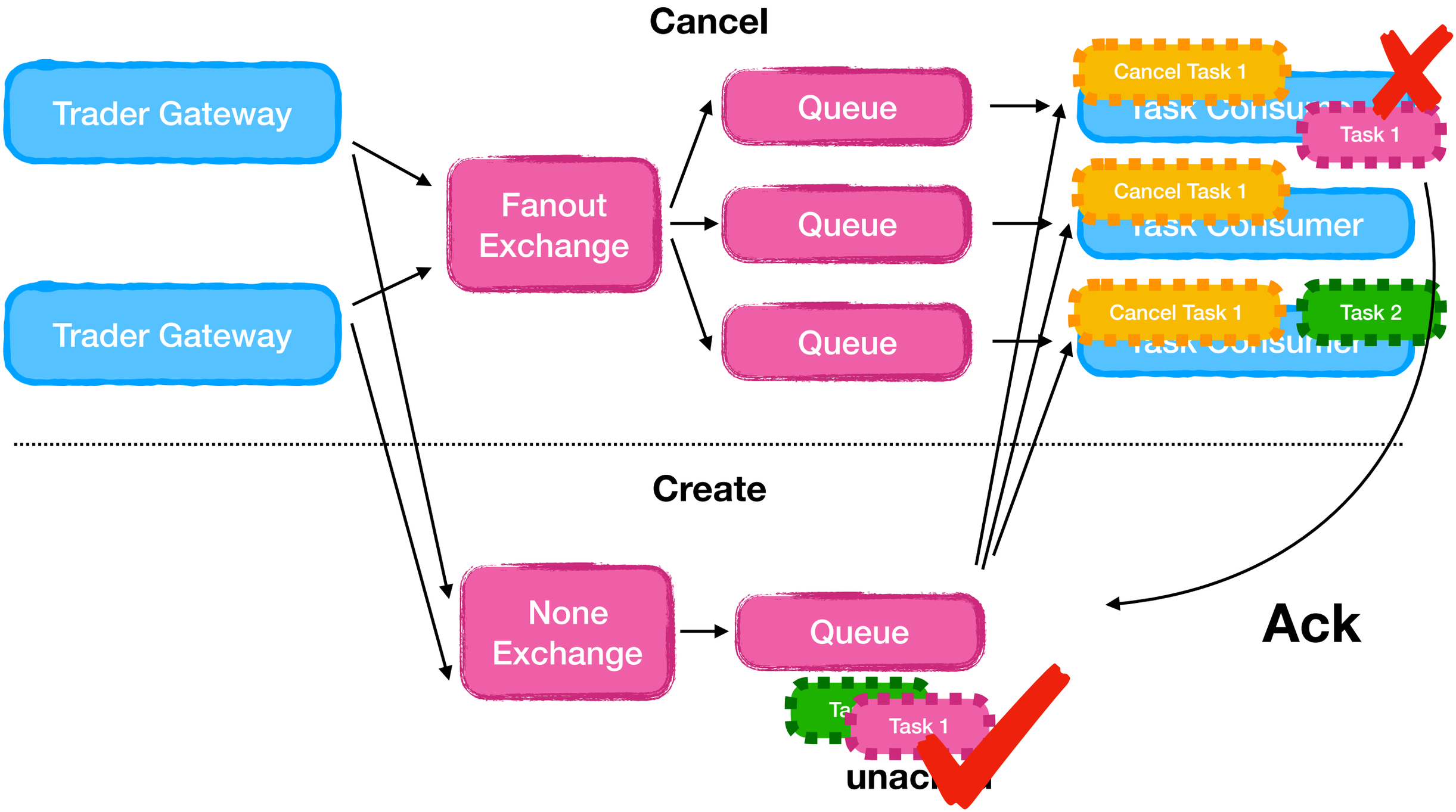
Situation 2 - fanout exchange



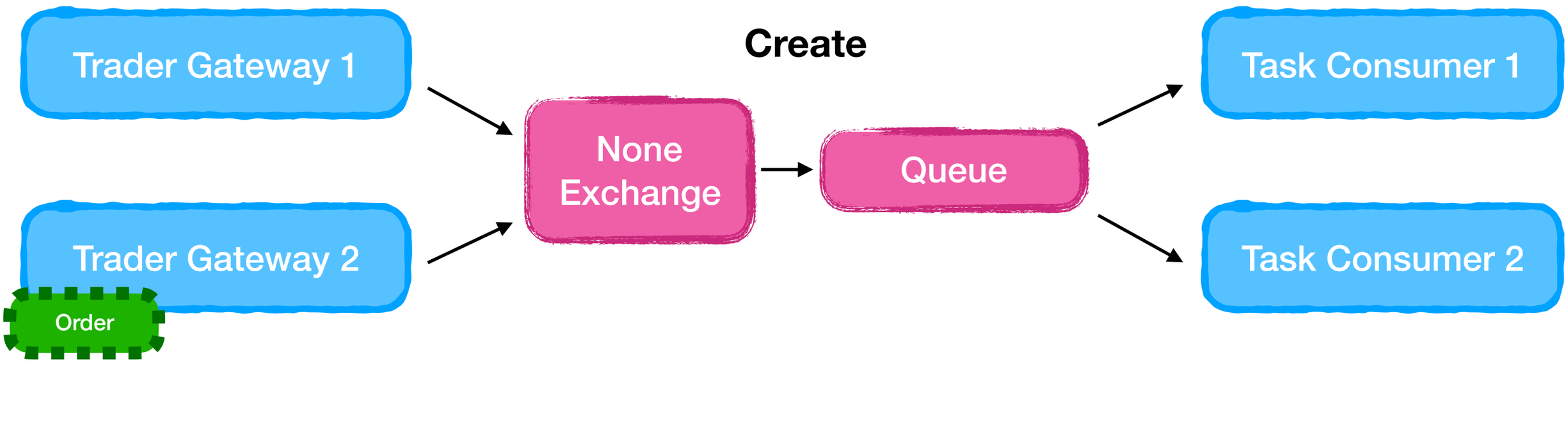
Situation 2 - fanout exchange



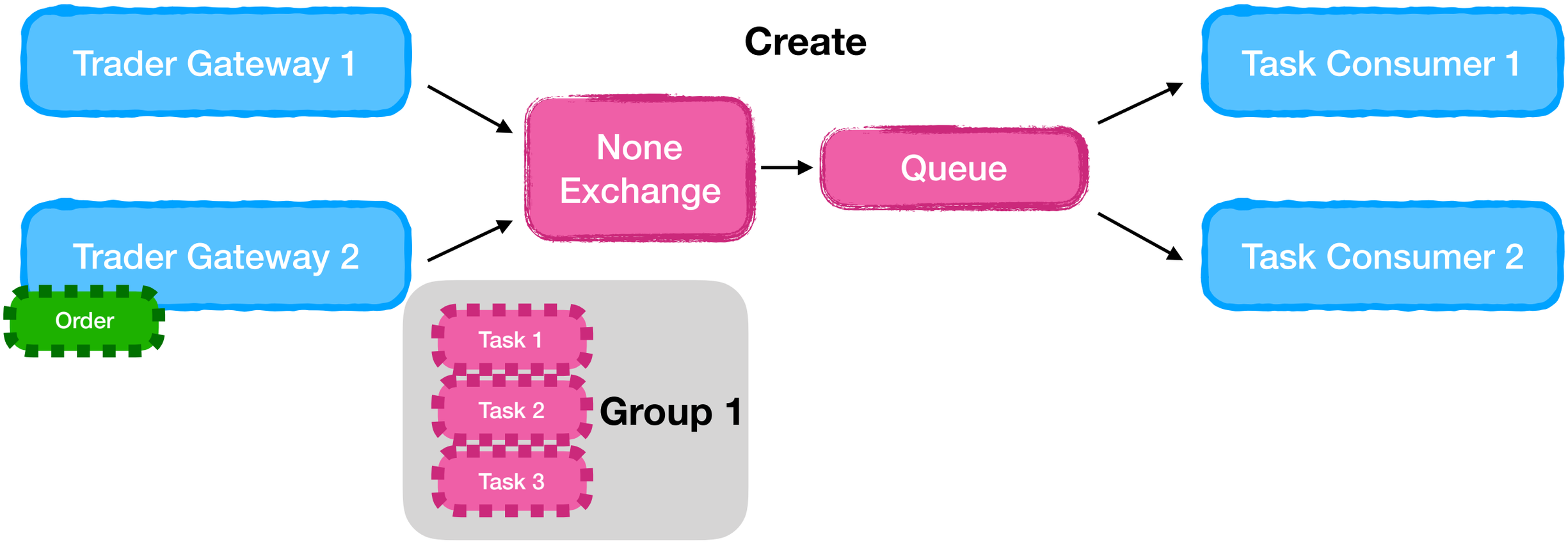
Situation 2 - fanout exchange



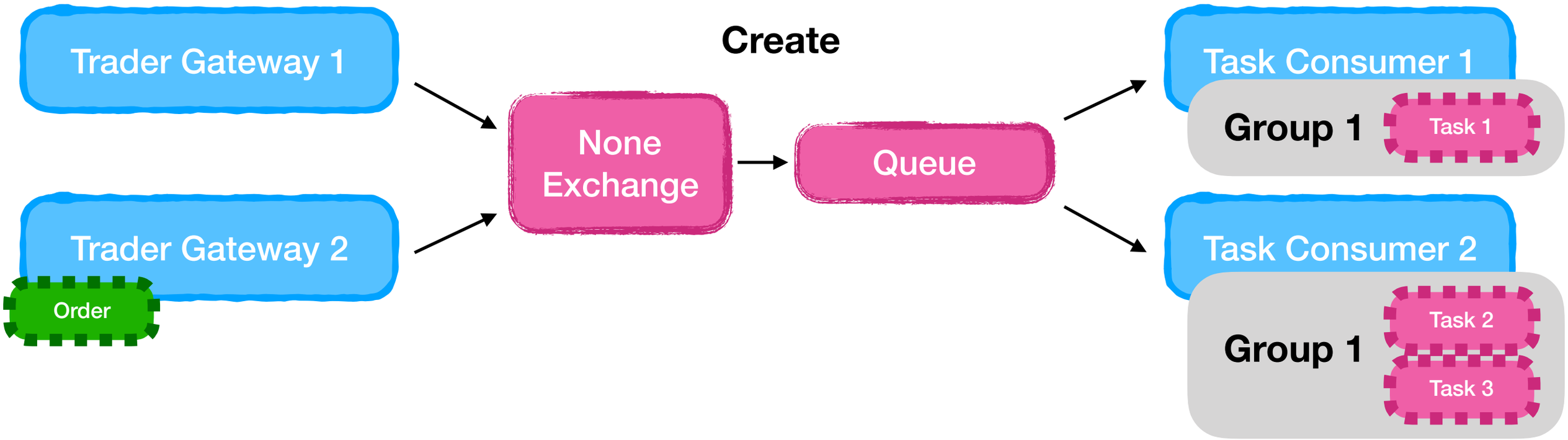
Situation 3 - more complex



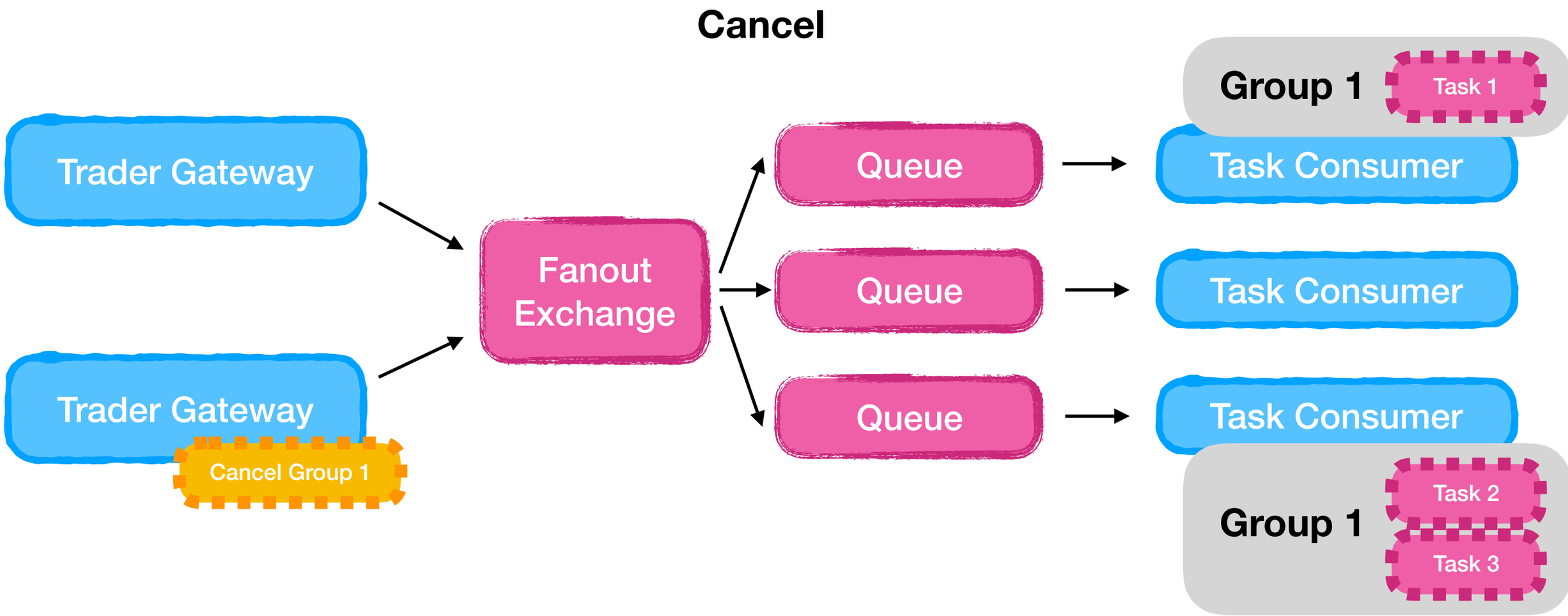
Situation 3 - more complex



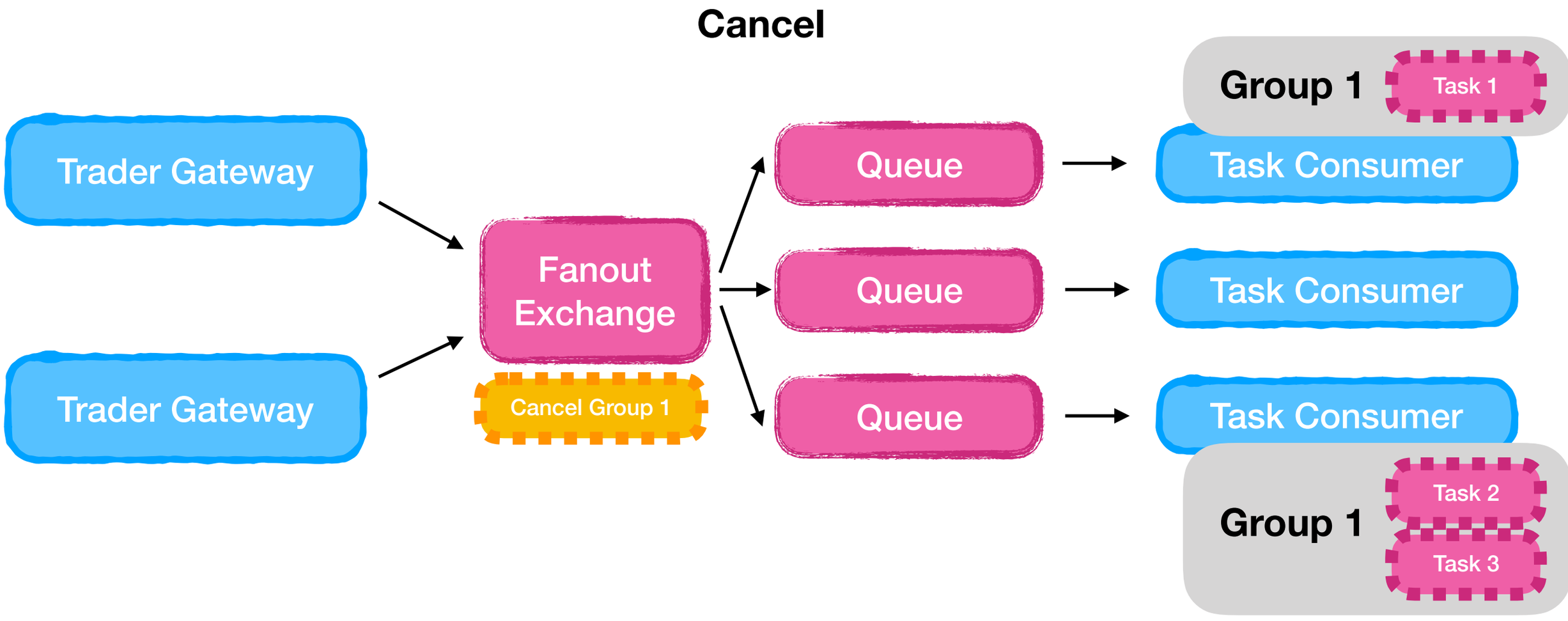
Situation 3 - more complex



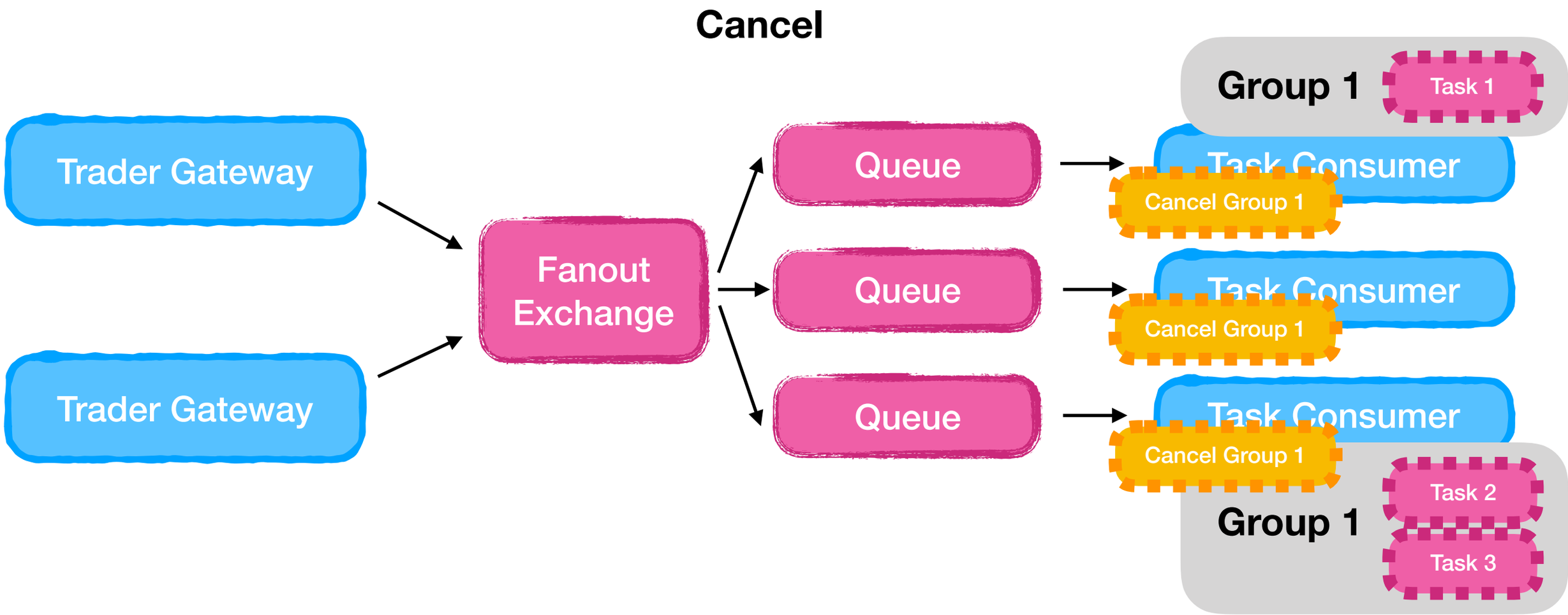
Situation 3 - more complex



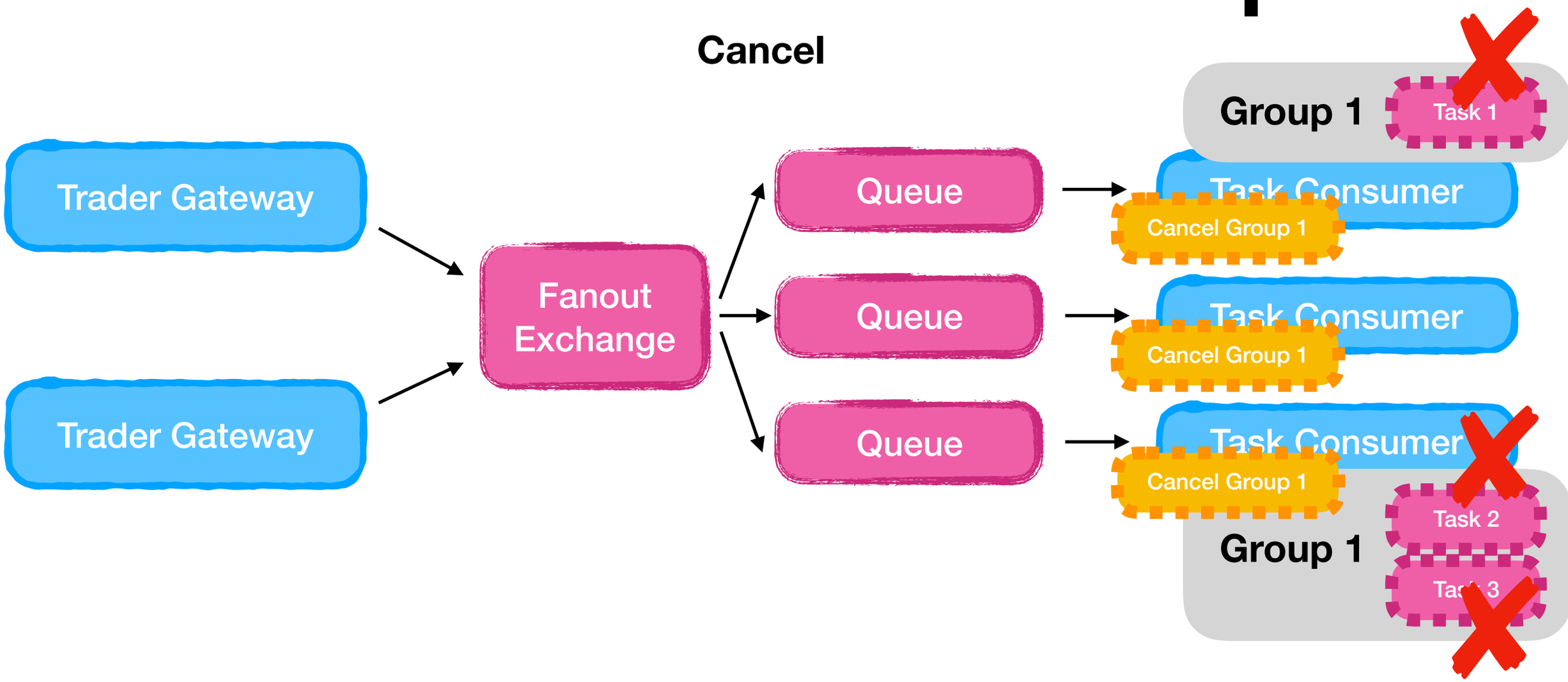
Situation 3 - more complex



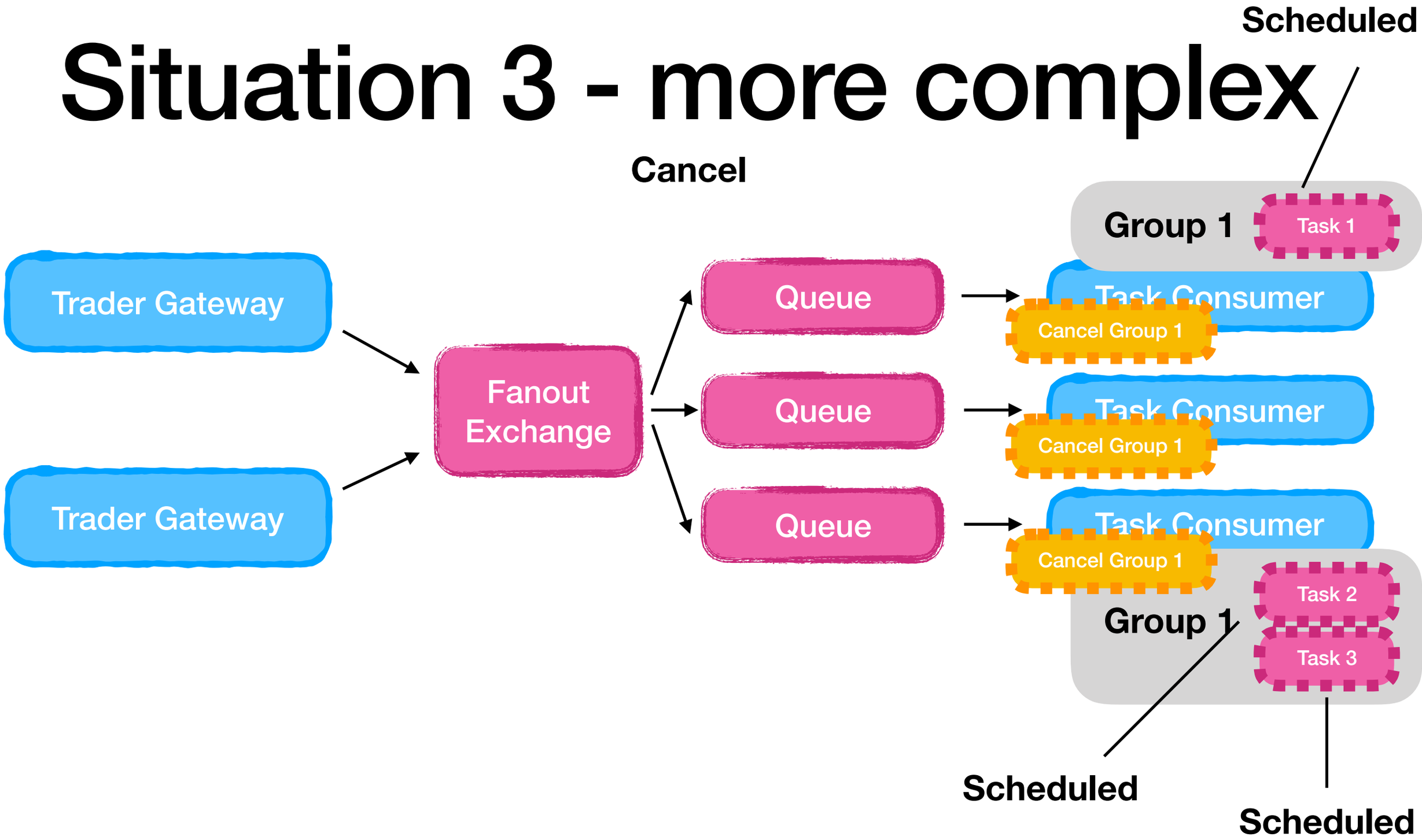
Situation 3 - more complex



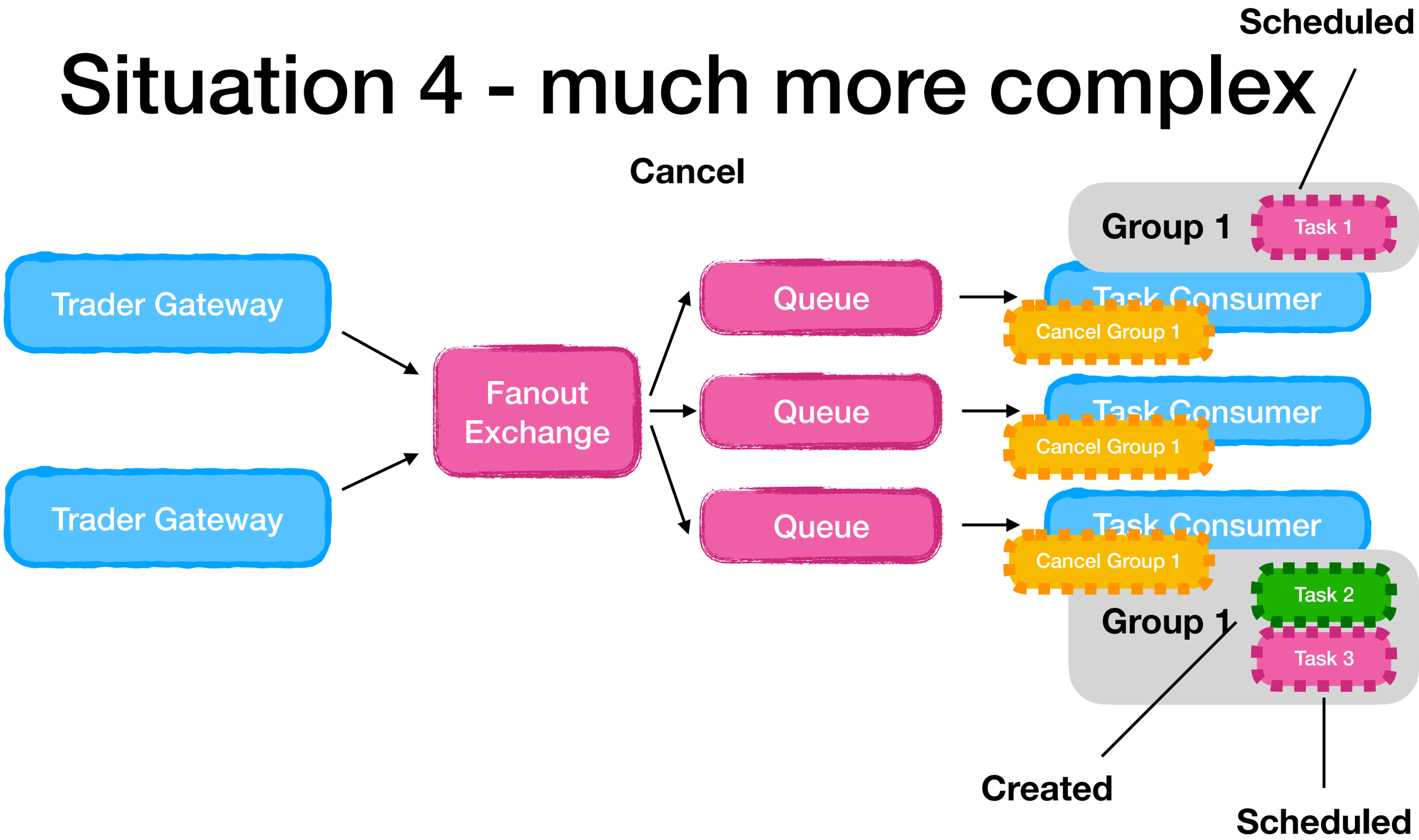
Situation 3 - more complex



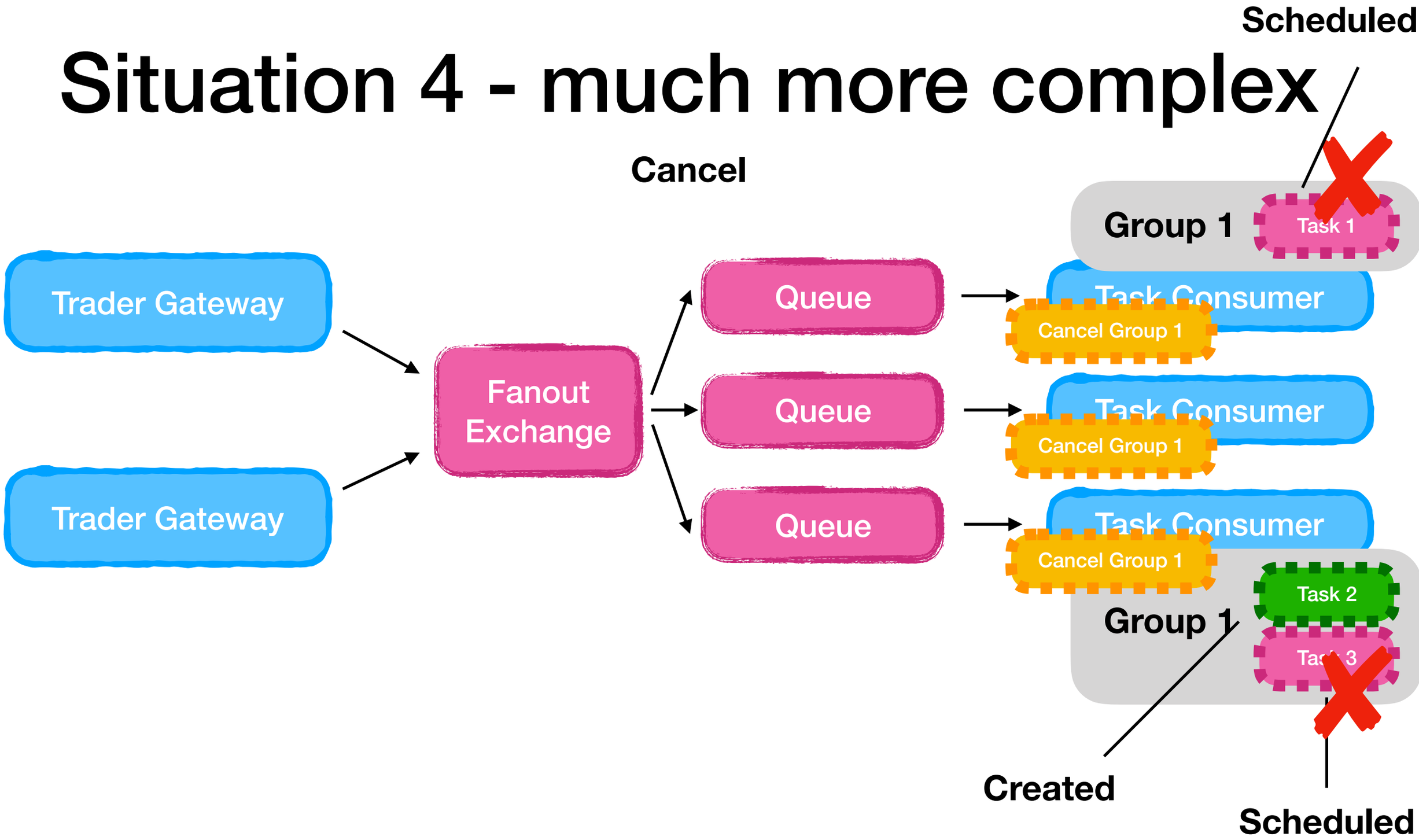
Situation 3 - more complex



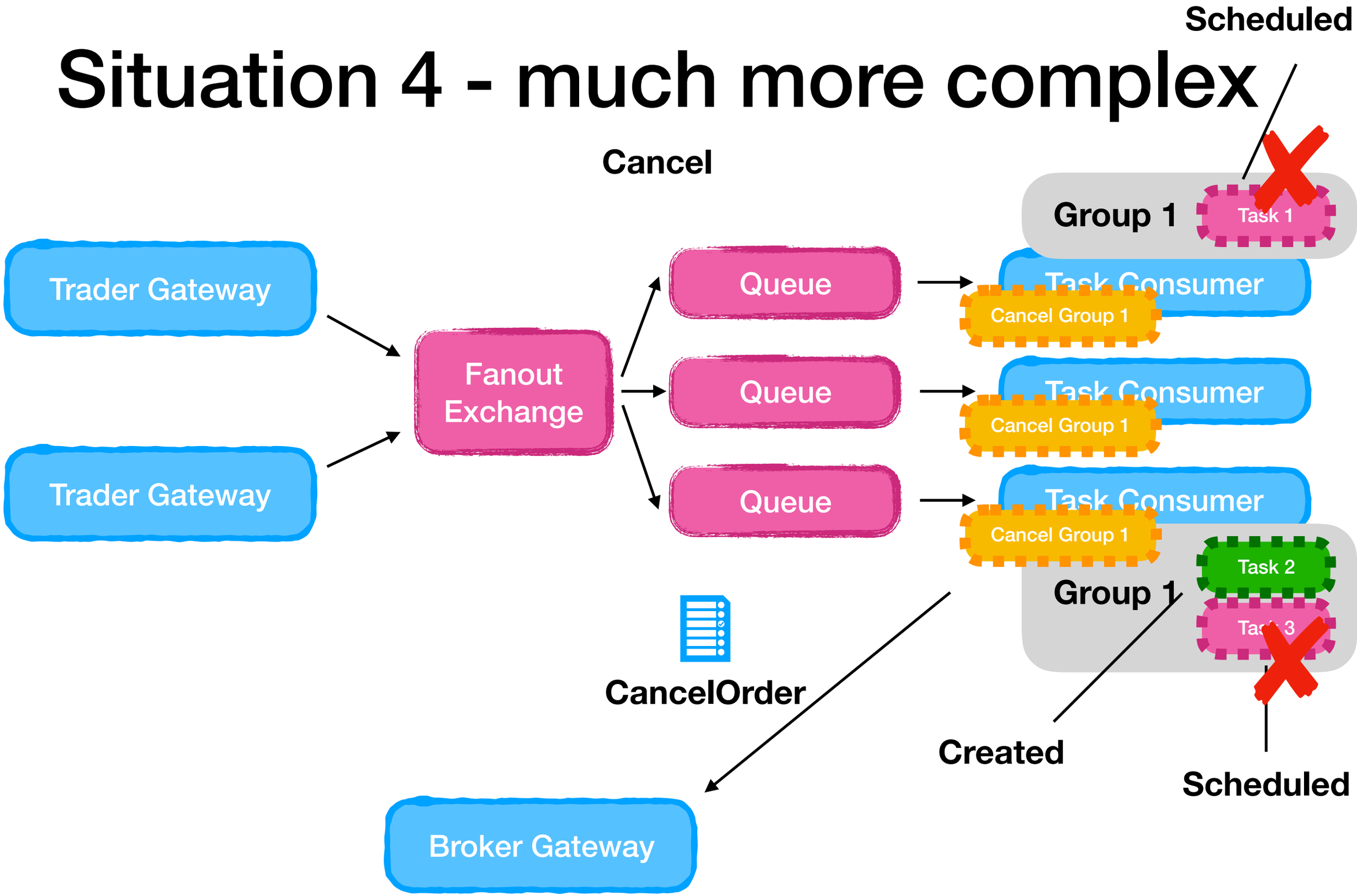
Situation 4 - much more complex



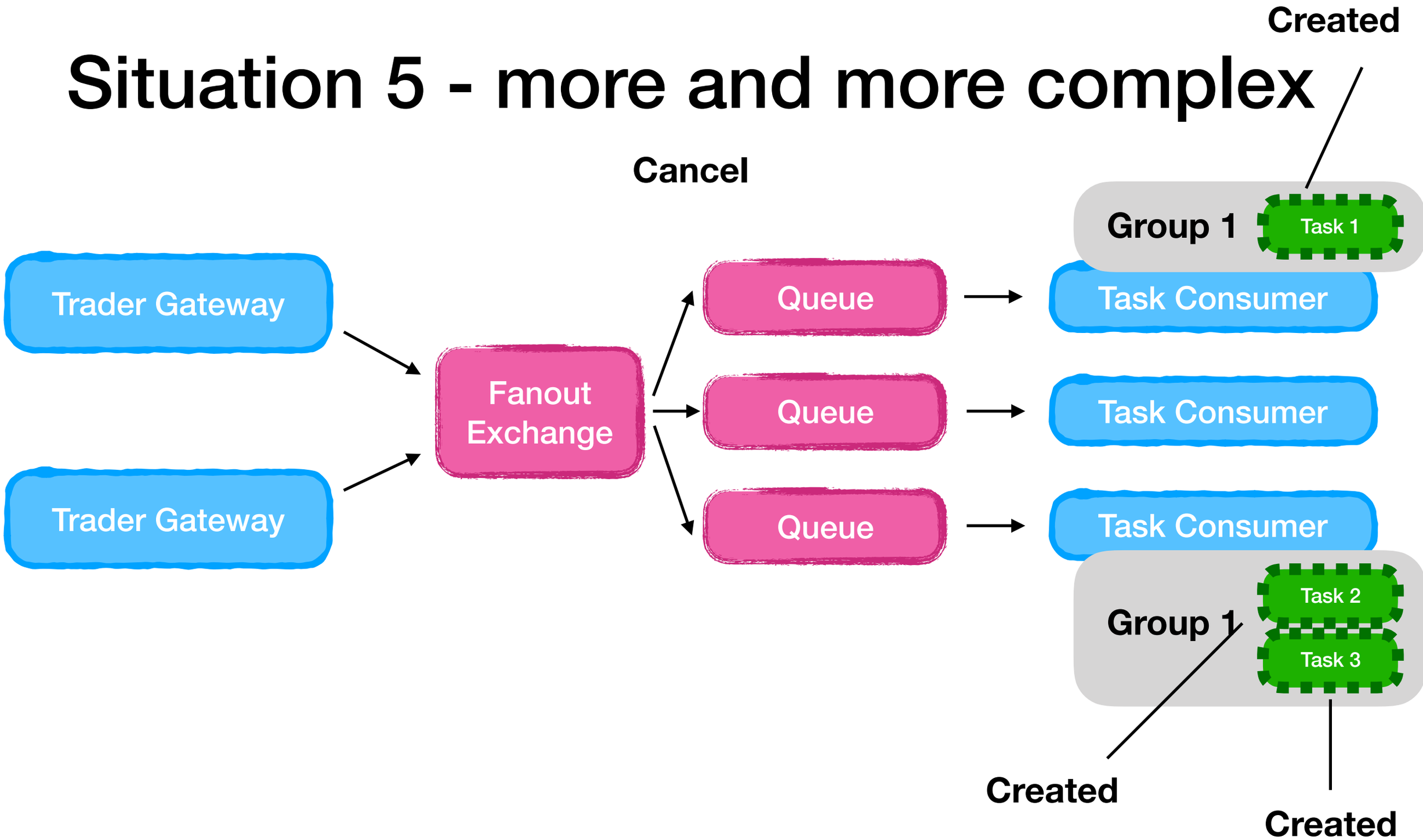
Situation 4 - much more complex



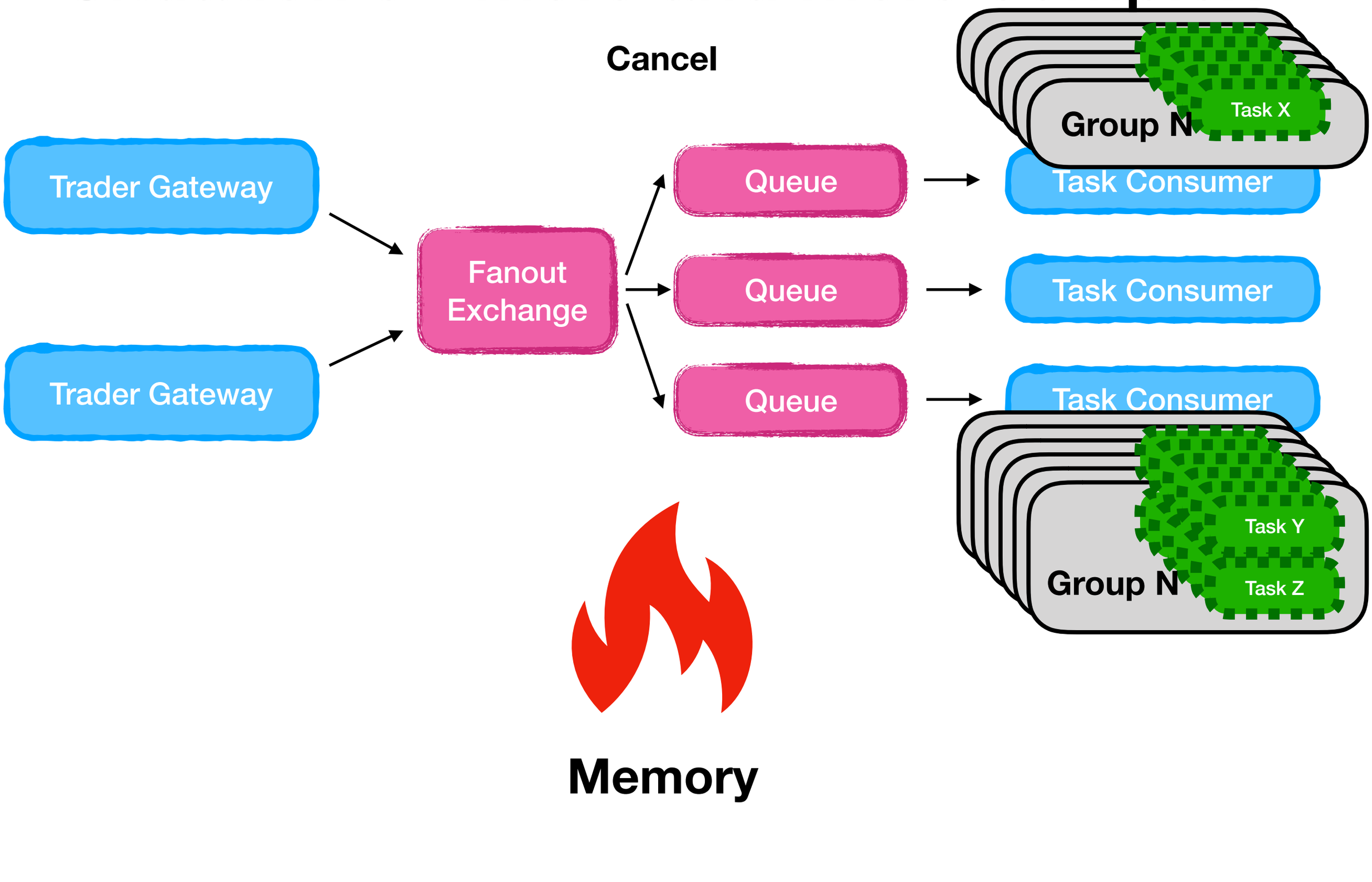
Situation 4 - much more complex



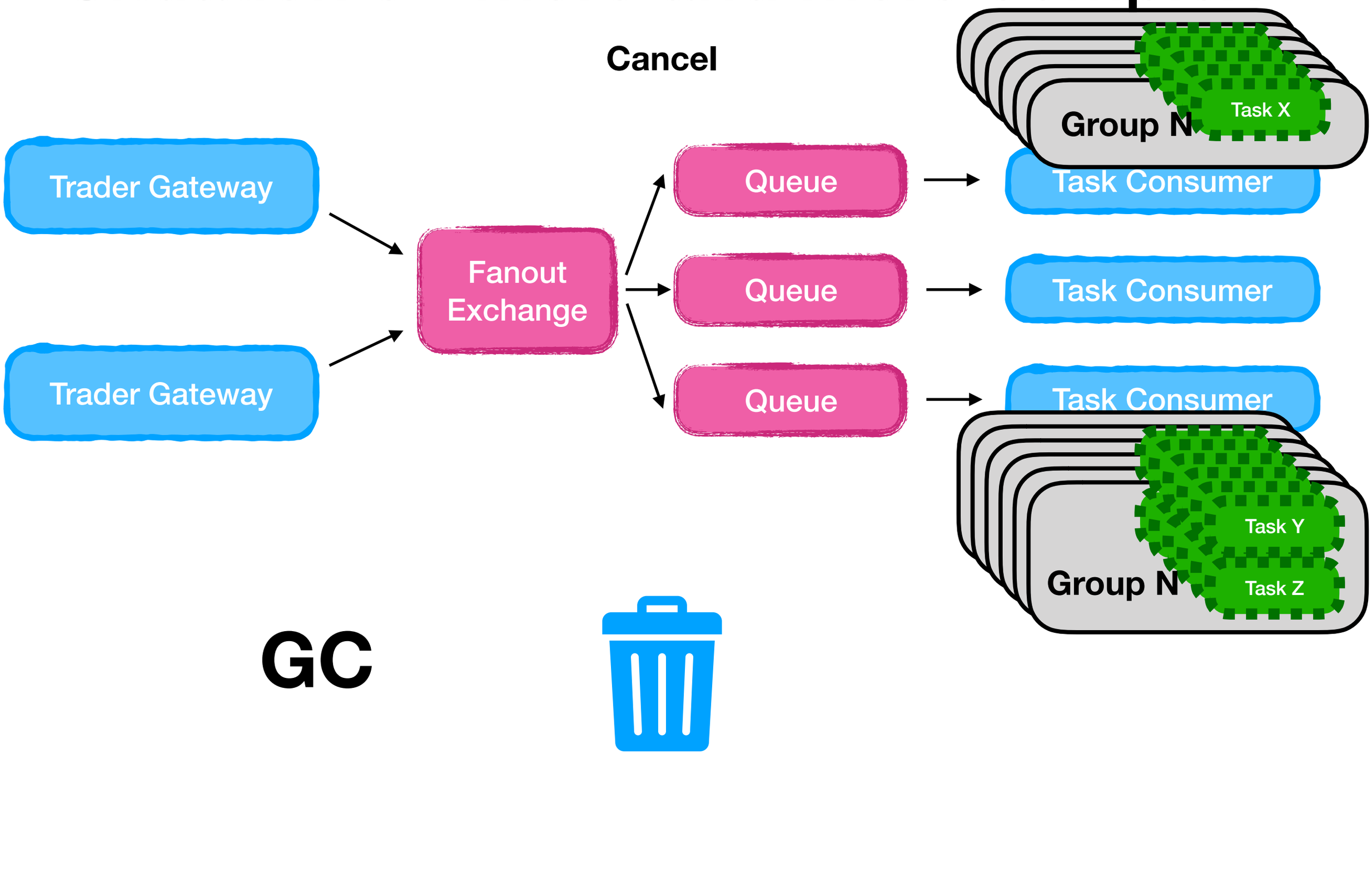
Situation 5 - more and more complex



Situation 5 - more and more complex

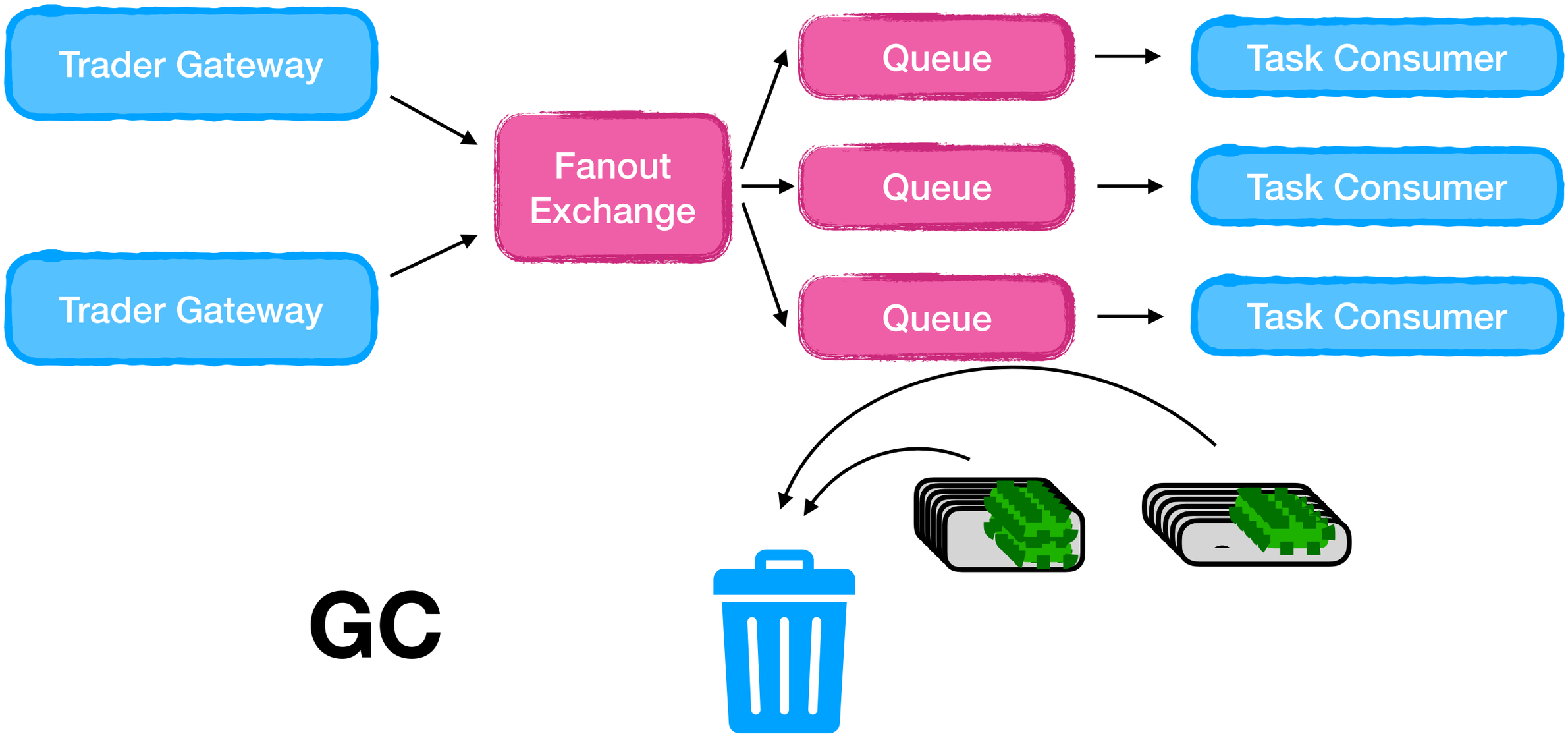


Situation 5 - more and more complex



Situation 5 - more and more complex

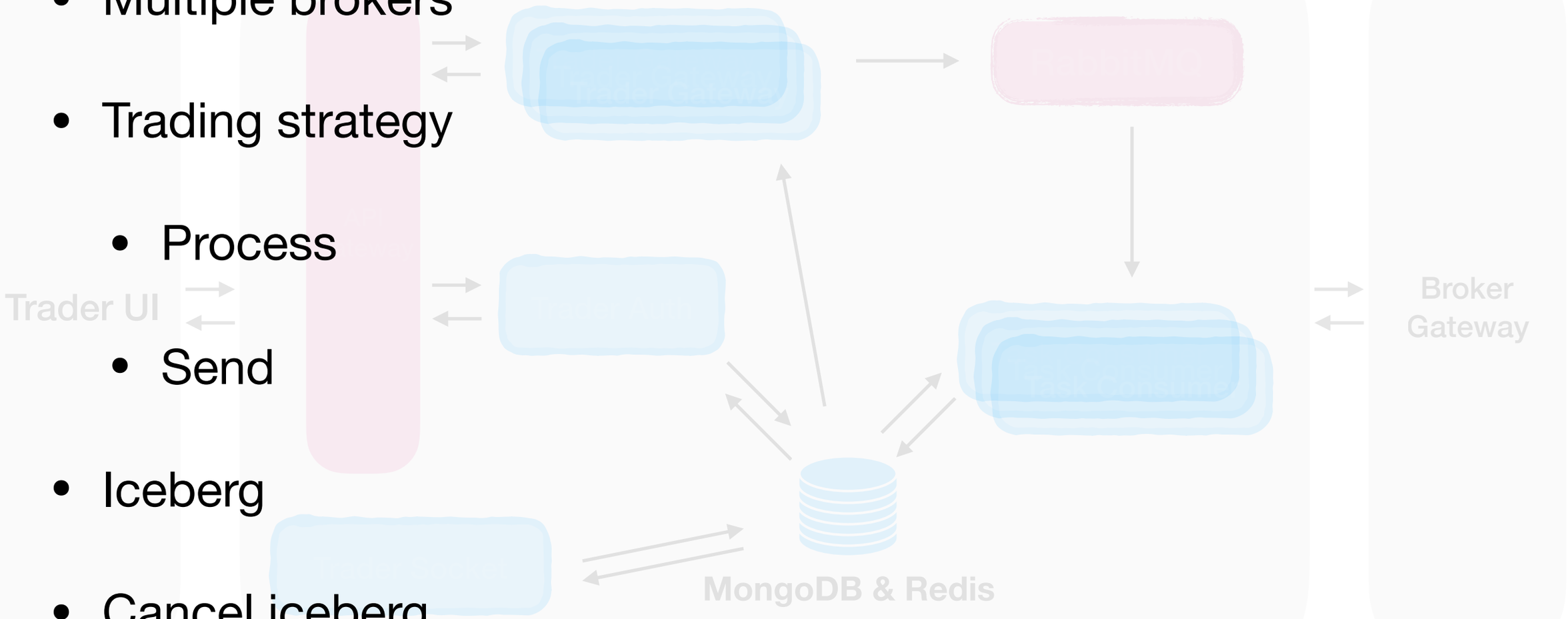
Cancel



GC

Summary

- Multiple brokers
- Trading strategy
    - Process
        - Send
- Iceberg
- Cancel iceberg
    - Both scheduled and created



Summary

- Multiple brokers
- Trading strategy
  - Process
    - Send
- Iceberg
- Cancel iceberg
  - Both scheduled and created

- One socket for one user
- Split the system
- Message Queue
  - [x] Duplicate consumption
  - [x] Message lost
- High availability

Trader Component