INFORMATICS INSTITUTE OF TECHNOLOGY

INFORMATICS INSTITUTE OF TECHNOLOGY

In Collaboration with

UNIVERSITY OF WESTMINSTER

Assessment Report

For

Inventory Management System

A development by

Nimesh Ekanayake (w1867890)

Course Work 2

Concurrent and **Distributed Systems**

(7SENG007C)

September 2022

Table of Contents

Introduction

The inventory management system is intended to manage the inventory of goods via a distributed system. Managers of warehouses are facilitated with highly available services to continue the daily warehouse order processing without any impact.

Specialty of the system

- Highly available
- High accuracy
- Fault tolerance ability

Target audience

- Clerks
- Warehouse managers

Pre-requisites

Project files

Extract the following zip files into a location and use the provided commands to run the project.

1. **IIT-Inventory-NameService.zip**
2. **IIT-Inventory-Server.zip**
3. **IIT-Inventory-Client.zip**

Example extraction location

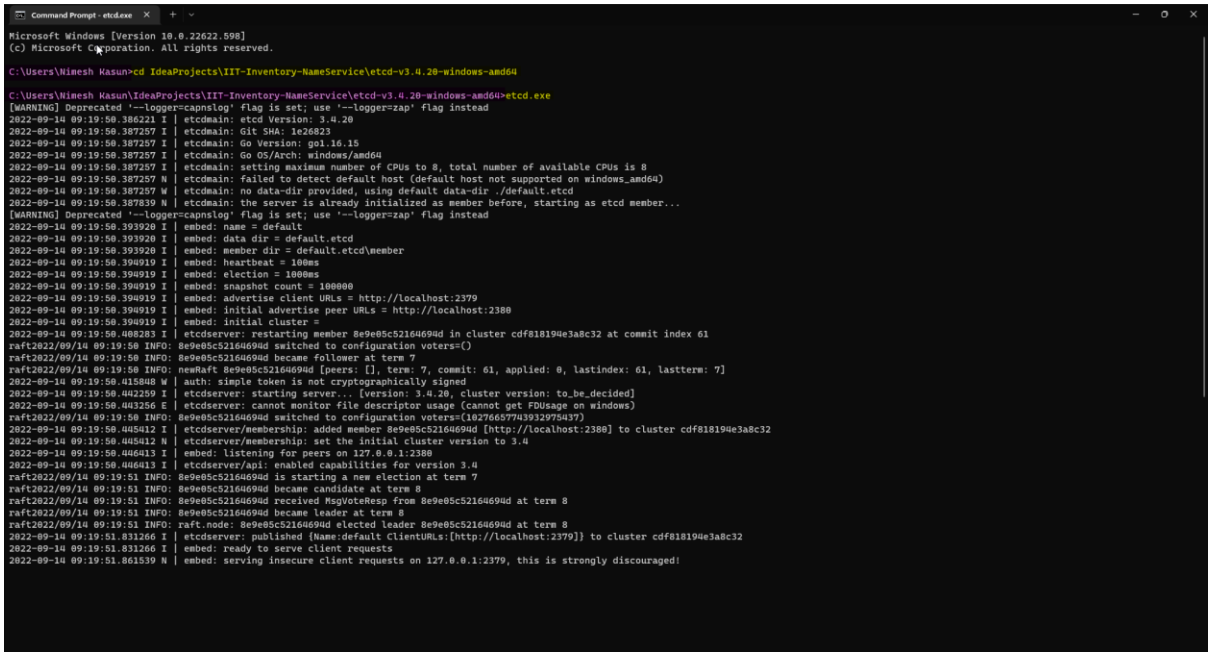
C:\Users*Nimesh Kasun*\IdeaProjects

Commands to run the servers (Windows)

Open Command Prompt/ Terminal

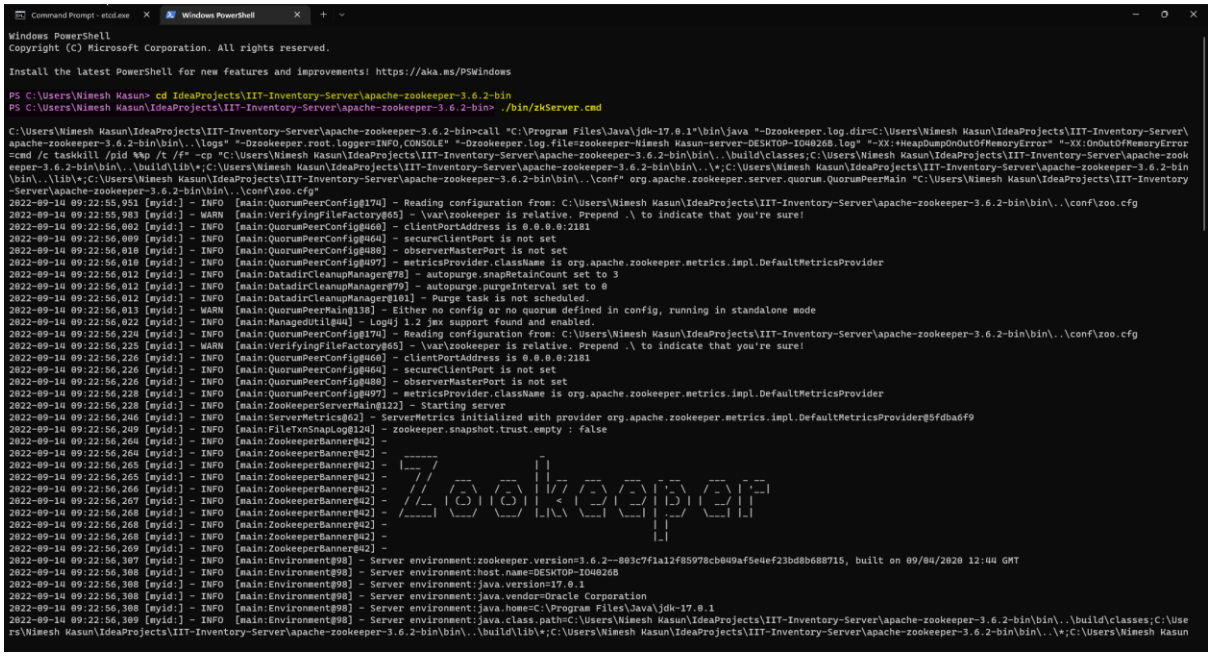
Start NameService Server:

```
> cd IdeaProjects\IIT-Inventory-NameService\etcd-v3.4.20-windows-amd64

> etcd.exe
```



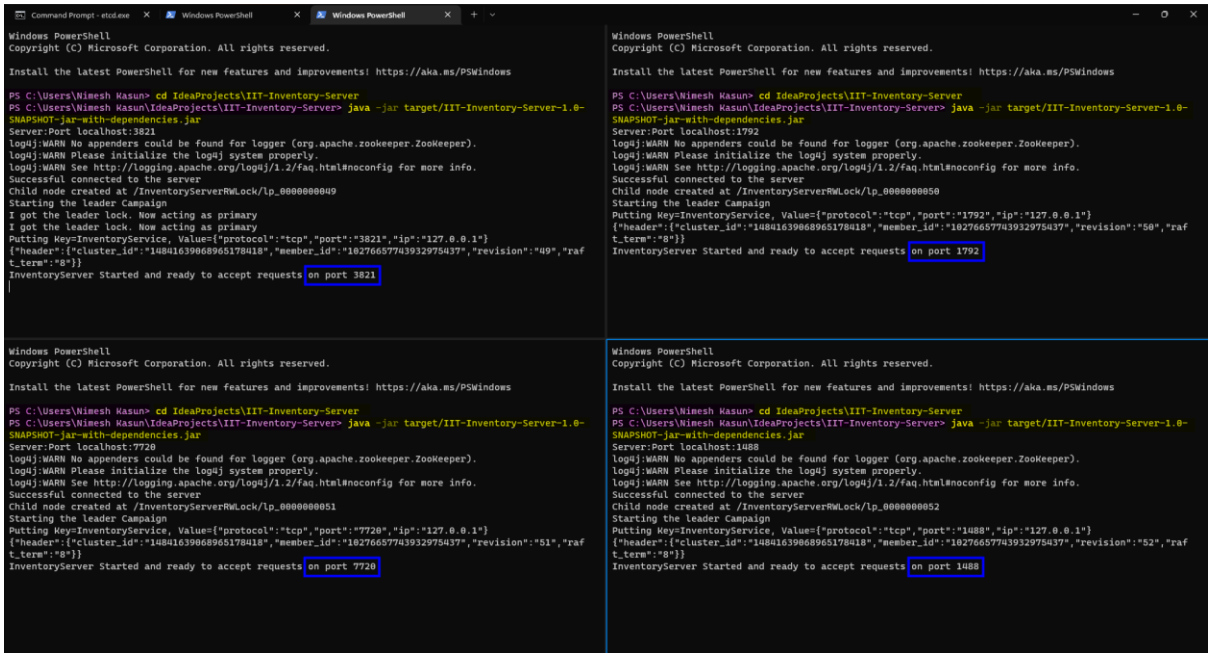
Start ZooKeeper Server:

```
> cd IdeaProjects\IIT-Inventory-Server\apache-zookeeper-3.6.2-bin

> ./bin/zkServer.cmd
```



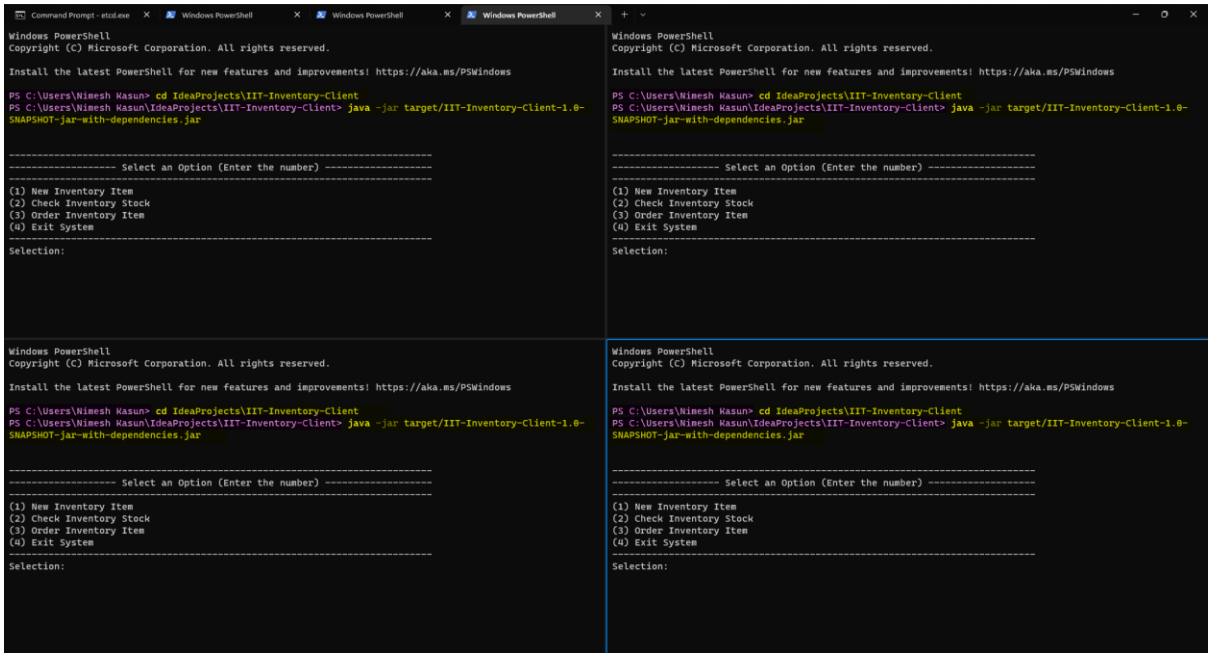
Start Inventory Server with multiple processes

```
> cd IdeaProjects\IIT-Inventory-Server

> java -jar target/IIT-Inventory-Server-1.0-SNAPSHOT-jar-with-dependencies.jar
```



Start Client Server as multiple warehouses

```
> cd IdeaProjects\IIT-Inventory-Client

> java -jar target/IIT-Inventory-Client-1.0-SNAPSHOT-jar-with-dependencies.jar
```



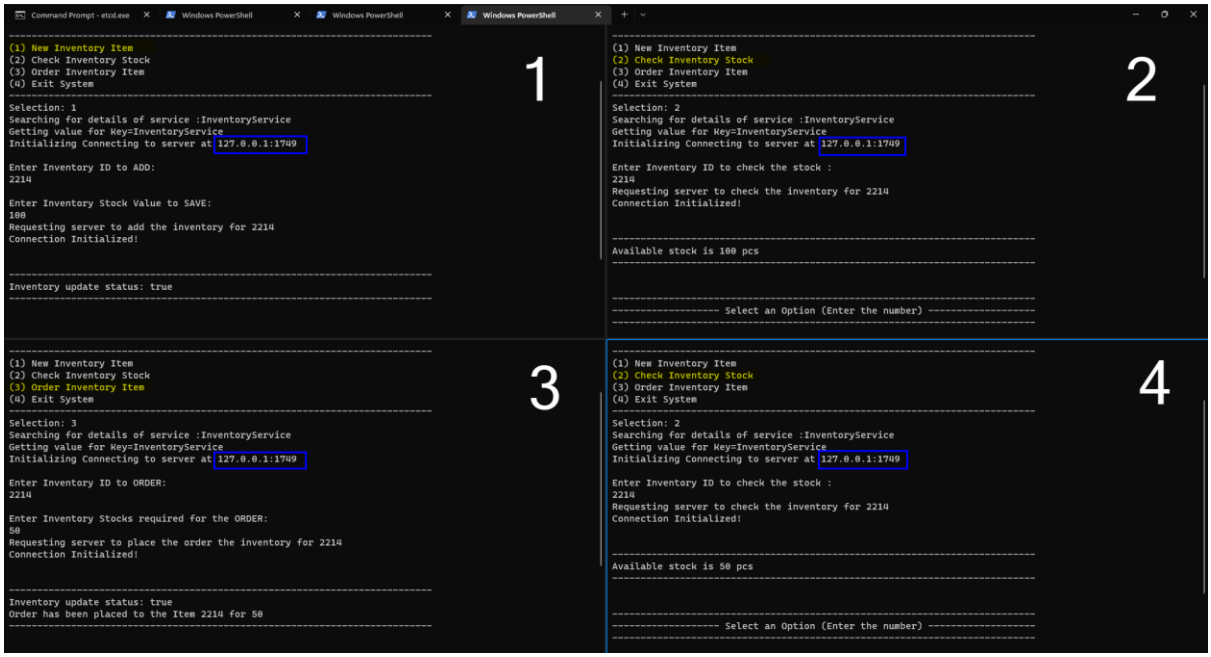
System Functionality

The developed system is integrated with ZooKeeper which allows to control one or more server nodes which are acting as multiple processes of the same server. These nodes are synchronized and in the event of unavailability of any note, other nodes are ready to serve the client requests without impacting the user experience or a loss of data.

Below explains the ZooKeeper integration used.

- Any number of client nodes can be used to add new stocks to the system.
    - If the item is new, a new item will be added to the system.
    - If the stock arrives for an existing item, system will be adding the new stock to the existing stock count for that item.

- Below picture shows the client interface and all clients has connected to the last started Inventory Server which is online via 127.0.0.1:**1749**

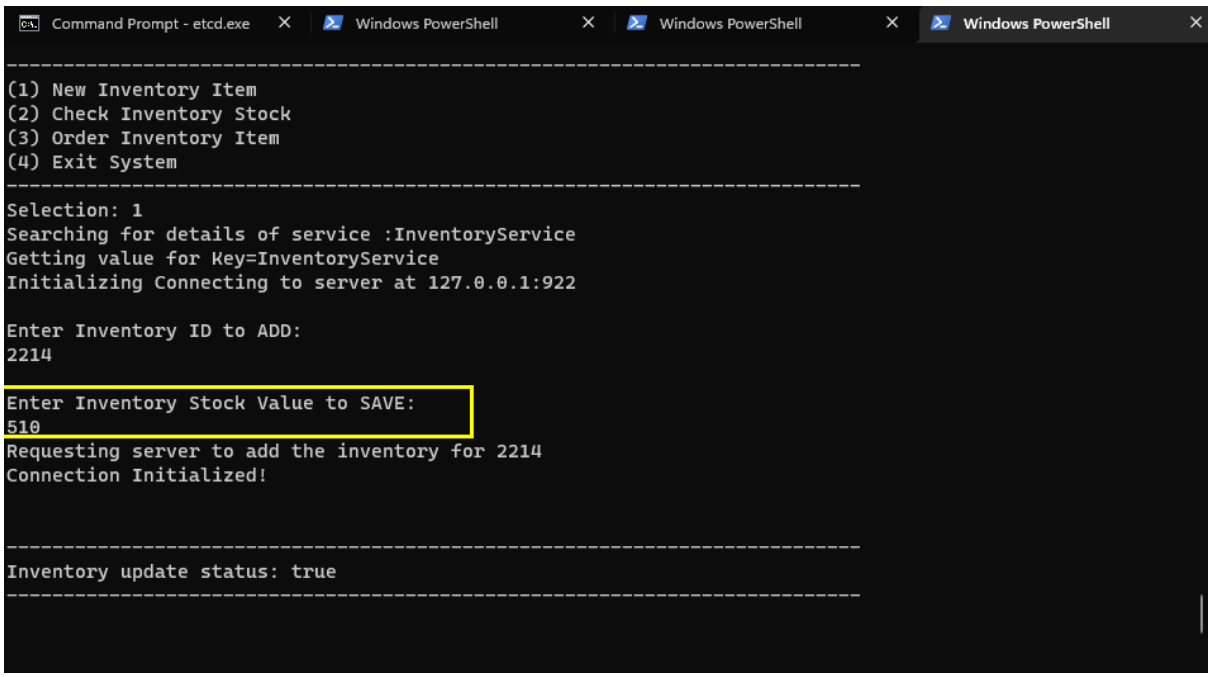


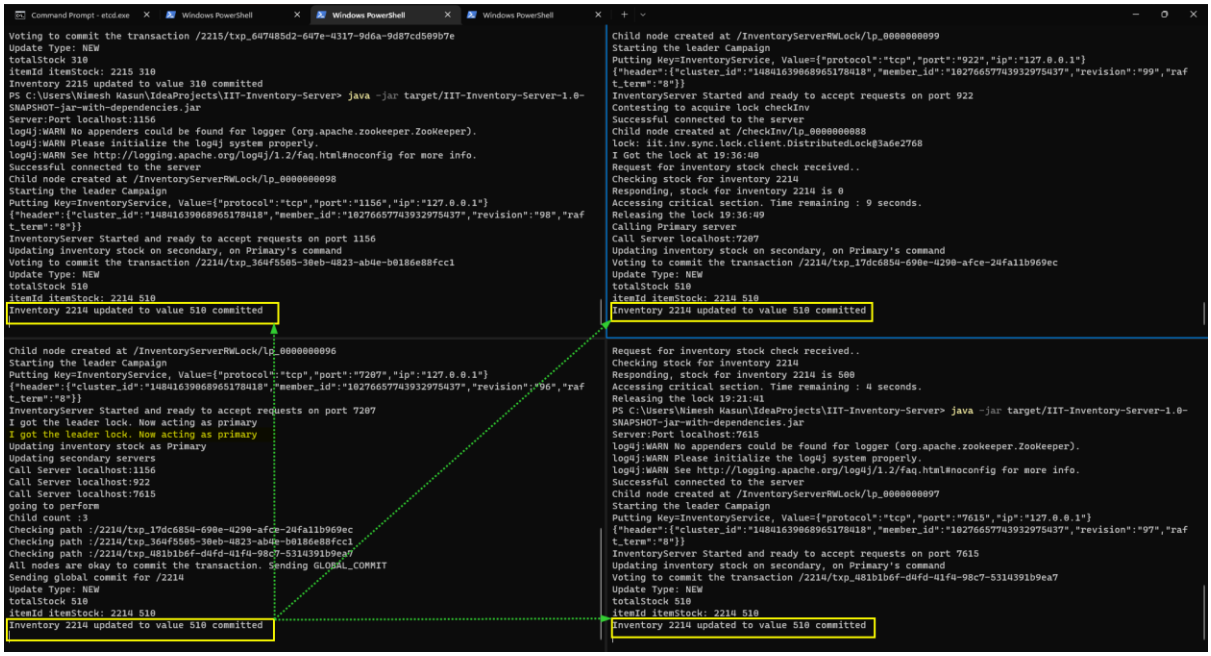
    - However, while executing the client requests, this server will be calling the primary server, which has the leader lock acquired. (127.0.0.1:**3991**)

- Below picture shows the multiple inventory server processes running on multiple ports.
    1. 127.0.0.1:**3991** (Got the leader lick since started first)
    2. 127.0.0.1:**2546**
    3. 127.0.0.1:**8957**
    4. 127.0.0.1:**1749** (Last started and all new clients are connected to this)

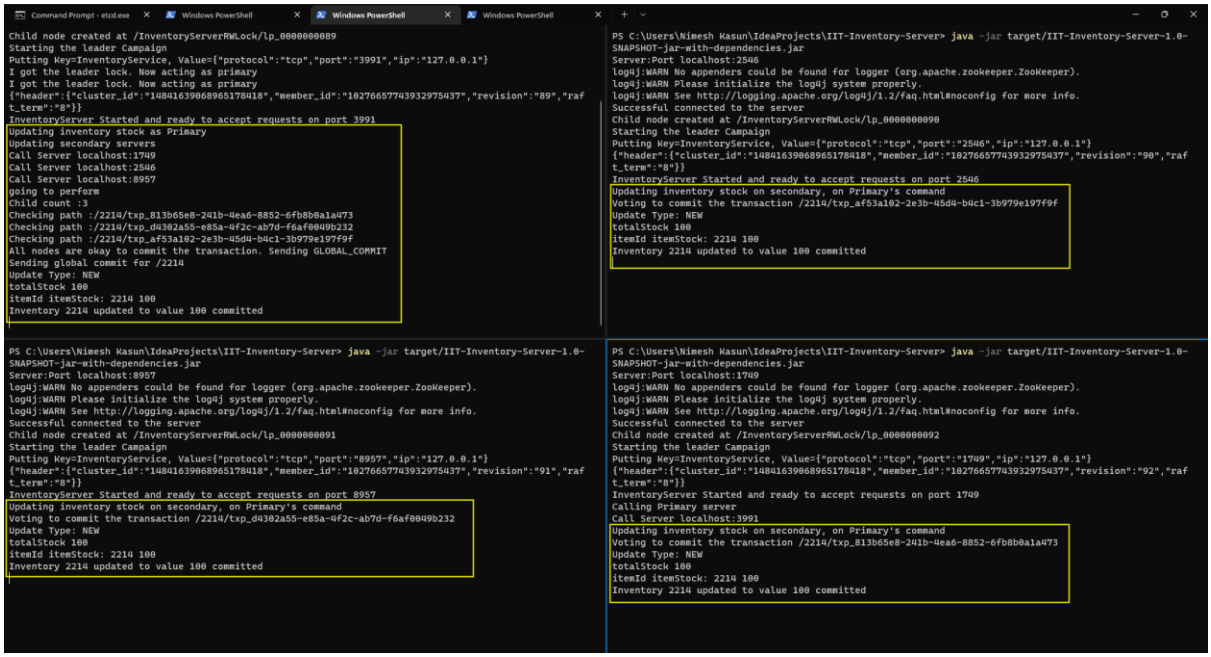
- Below set of images shows in order, the server logs when client requests followings in order:
  (1) New Inventory Item *(Image 1 below)*
  (2) Check Inventory Item *(Image 2 below)*
  (3) Order Inventory Item *(Image 3 below)*
  (4) Check Inventory Item *(Image 4 below)*

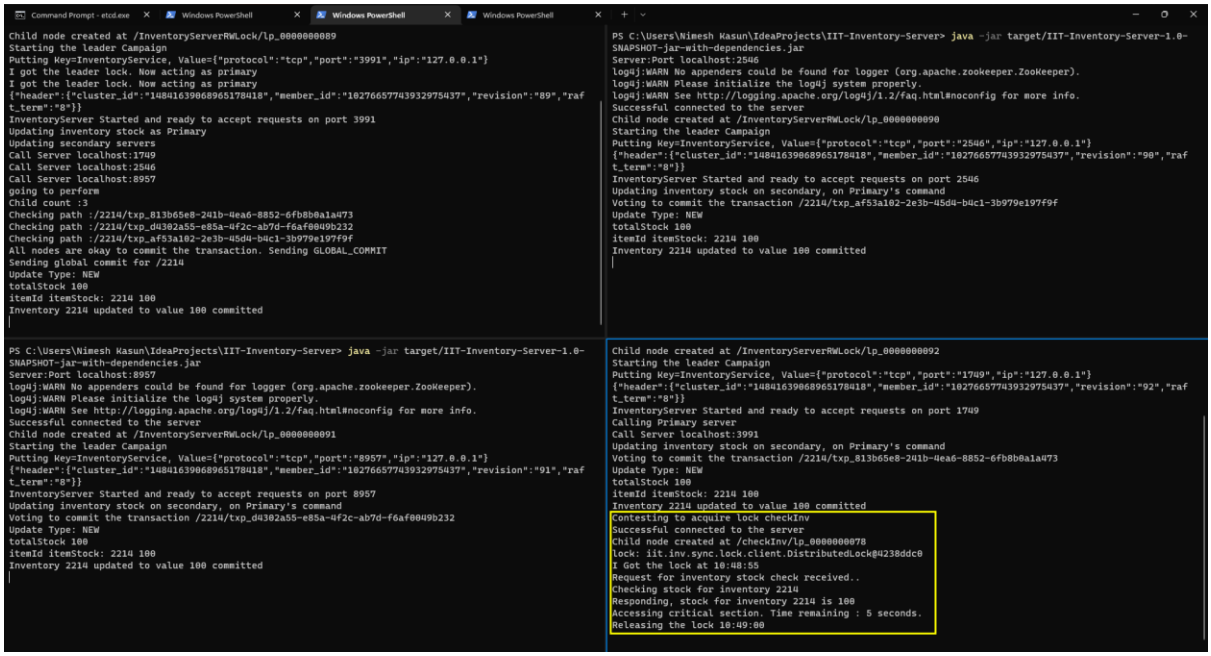
    - Based on below images, it's noticeable that only the primary server who got the leader lock is processing requests and then notify all other secondary servers/nodes to update their data as well.

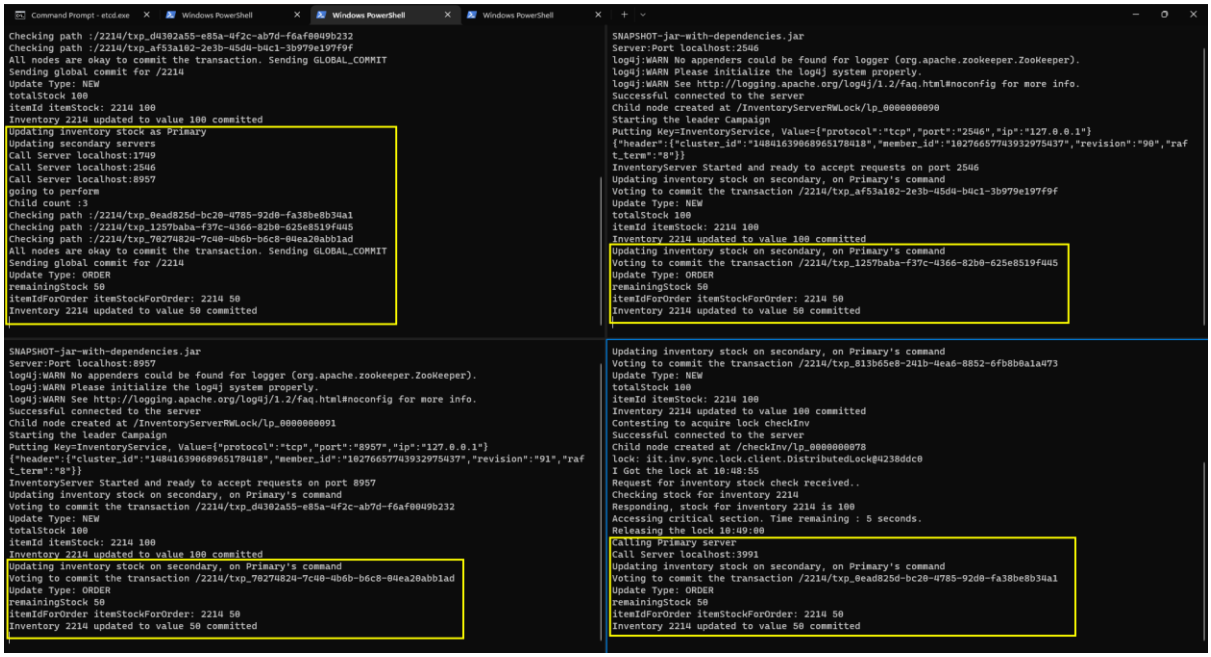


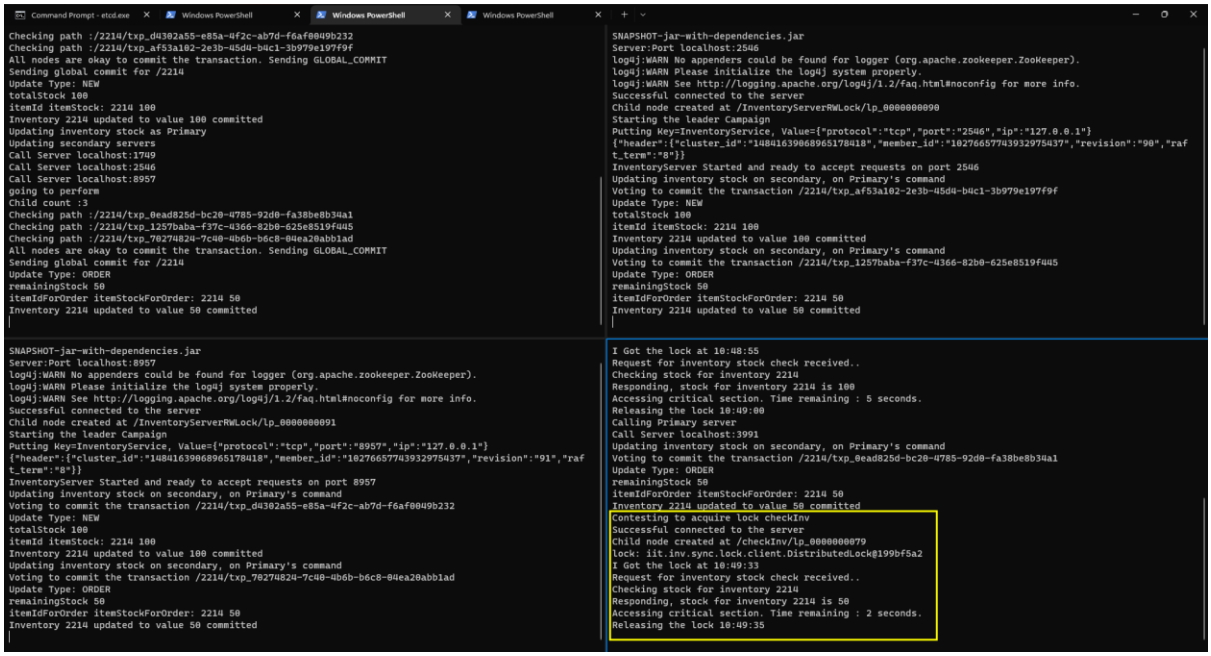


- This enables all the servers to maintain the accuracy and be ready in case the primary server shutdown, then to get the leadership as assigned and act as the new primary server.

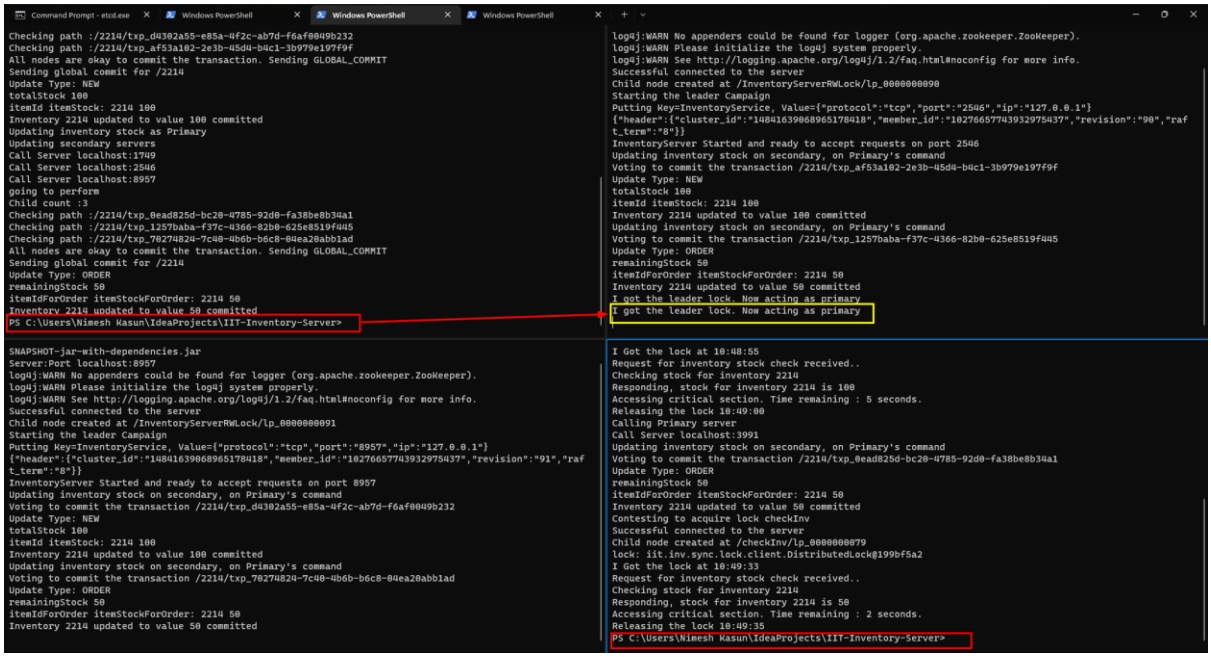


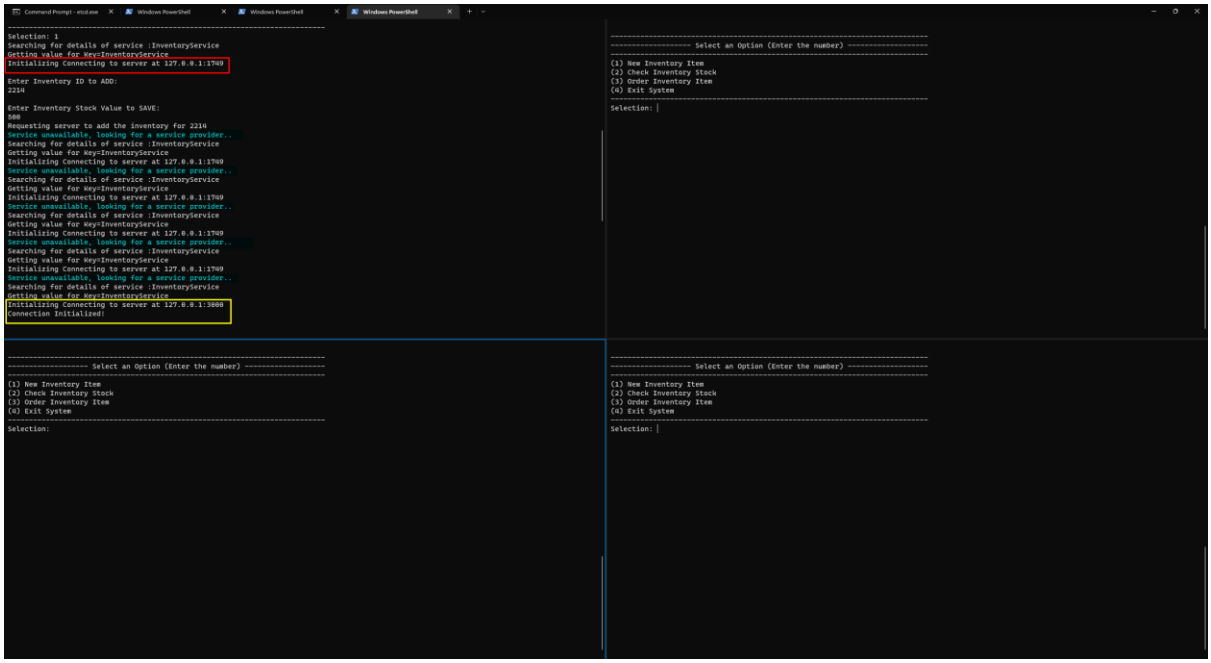

Child node created at /InventoryServerRWLock/lp_0000000089
Starting the leader Campaign
Putting Key=InventoryService, Value={"protocol":"tcp","port":"3991","ip":"127.0.0.1"}
I got the leader lock. Now acting as primary
I got the leader lock. Now acting as primary
{"header":{"cluster_id":"14841639068965178418","member_id":"10276657743932975437","revision":"89","raft_term":"8"}}
InventoryServer Started and ready to accept requests on port 3991
Updating inventory stock as Primary
Updating secondary servers
Call Server localhost:1749
Call Server localhost:2546
Call Server localhost:8957
going to perform
Child count :3
Checking path :/2214/txp_813b65e8-241b-4ea6-8852-6fb8b0a1a473
Checking path :/2214/txp_d4302a55-e85a-4f2c-ab7d-f6af0049b232
Checking path :/2214/txp_af53a102-2e3b-45d4-b4c1-3b979e197f9f
All nodes are okay to commit the transaction. Sending GLOBAL_COMMIT
Sending global commit for /2214
Update Type: NEW
totalStock 100
itemId itemStock: 2214 100
Inventory 2214 updated to value 100 committed
PS C:\Users\Nimesh Kasun\IdeaProjects\IIT-Inventory-Server> java -jar target/IIT-Inventory-Server-1.0-SNAPSHOT-jar-with-dependencies.jar
Server:Port localhost:2546
log4j:WARN No appenders could be found for logger (org.apache.zookeeper.ZooKeeper).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Successful connected to the server
Child node created at /InventoryServerRWLock/lp_0000000090
Starting the leader Campaign
Putting Key=InventoryService, Value={"protocol":"tcp","port":"2546","ip":"127.0.0.1"}
{"header":{"cluster_id":"14841639068965178418","member_id":"10276657743932975437","revision":"90","raft_term":"8"}}
InventoryServer Started and ready to accept requests on port 2546
Updating inventory stock on secondary, on Primary's command
Voting to commit the transaction /2214/txp_af53a102-2e3b-45d4-b4c1-3b979e197f9f
Update Type: NEW
totalStock 100
itemId itemStock: 2214 100
Inventory 2214 updated to value 100 committed
PS C:\Users\Nimesh Kasun\IdeaProjects\IIT-Inventory-Server> java -jar target/IIT-Inventory-Server-1.0-SNAPSHOT-jar-with-dependencies.jar
Server:Port localhost:8957
log4j:WARN No appenders could be found for logger (org.apache.zookeeper.ZooKeeper).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Successful connected to the server
Child node created at /InventoryServerRWLock/lp_0000000091
Starting the leader Campaign
Putting Key=InventoryService, Value={"protocol":"tcp","port":"8957","ip":"127.0.0.1"}
{"header":{"cluster_id":"14841639068965178418","member_id":"10276657743932975437","revision":"91","raft_term":"8"}}
InventoryServer Started and ready to accept requests on port 8957
Updating inventory stock on secondary, on Primary's command
Voting to commit the transaction /2214/txp_d4302a55-e85a-4f2c-ab7d-f6af0049b232
Update Type: NEW
totalStock 100
itemId itemStock: 2214 100
Inventory 2214 updated to value 100 committed
Child node created at /InventoryServerRWLock/lp_0000000092
Starting the leader Campaign
Putting Key=InventoryService, Value={"protocol":"tcp","port":"1749","ip":"127.0.0.1"}
{"header":{"cluster_id":"14841639068965178418","member_id":"10276657743932975437","revision":"92","raft_term":"8"}}
InventoryServer Started and ready to accept requests on port 1749
Calling Primary server
Call Server localhost:3991
Updating inventory stock on secondary, on Primary's command
Voting to commit the transaction /2214/txp_813b65e8-241b-4ea6-8852-6fb8b0a1a473
Update Type: NEW
totalStock 100
itemId itemStock: 2214 100
Inventory 2214 updated to value 100 committed
Contesting to acquire lock checkInv
Successful connected to the server
Child node created at /checkInv/lp_0000000078
lock: iit.inv.sync.lock.client.DistributedLock@4238ddc0
I Got the lock at 10:48:55
Request for inventory stock check received..
Checking stock for inventory 2214
Responding, stock for inventory 2214 is 100
Accessing critical section. Time remaining : 5 seconds.
Releasing the lock 10:49:00


Checking path :/2214/txp_d4302a55-e85a-4f2c-ab7d-f6af0049b232
Checking path :/2214/txp_af53a102-2e3b-45d4-b4c1-3b979e197f9f
All nodes are okay to commit the transaction. Sending GLOBAL_COMMIT
Sending global commit for /2214
Update Type: NEW
totalStock 100
itemId itemStock: 2214 100
Inventory 2214 updated to value 100 committed
Updating inventory stock as Primary
Updating secondary servers
Call Server localhost:1749
Call Server localhost:2546
Call Server localhost:8957
going to perform
Child count :3
Checking path :/2214/txp_0ead825d-bc20-4785-92d0-fa38be8b34a1
Checking path :/2214/txp_1257baba-f37c-4366-82b0-625e8519f445
Checking path :/2214/txp_70274824-7c40-4b6b-b6c8-04ea20abb1ad
All nodes are okay to commit the transaction. Sending GLOBAL_COMMIT
Sending global commit for /2214
Update Type: ORDER
remainingStock 50
itemIdForOrder itemStockForOrder: 2214 50
Inventory 2214 updated to value 50 committed
SNAPSHOT-jar-with-dependencies.jar
Server:Port localhost:8957
log4j:WARN No appenders could be found for logger (org.apache.zookeeper.ZooKeeper).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Successful connected to the server
Child node created at /InventoryServerRWLock/lp_0000000091
Starting the leader Campaign
Putting Key=InventoryService, Value={"protocol":"tcp","port":"8957","ip":"127.0.0.1"}
{"header":{"cluster_id":"14841639068965178418","member_id":"10276657743932975437","revision":"91","raft_term":"8"}}
InventoryServer Started and ready to accept requests on port 8957
Updating inventory stock on secondary, on Primary's command
Voting to commit the transaction /2214/txp_d4302a55-e85a-4f2c-ab7d-f6af0049b232
Update Type: NEW
totalStock 100
itemId itemStock: 2214 100
Inventory 2214 updated to value 100 committed
Updating inventory stock on secondary, on Primary's command
Voting to commit the transaction /2214/txp_70274824-7c40-4b6b-b6c8-04ea20abb1ad
Update Type: ORDER
remainingStock 50
itemIdForOrder itemStockForOrder: 2214 50
Inventory 2214 updated to value 50 committed
SNAPSHOT-jar-with-dependencies.jar
Server:Port localhost:2546
log4j:WARN No appenders could be found for logger (org.apache.zookeeper.ZooKeeper).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Successful connected to the server
Child node created at /InventoryServerRWLock/lp_0000000090
Starting the leader Campaign
Putting Key=InventoryService, Value={"protocol":"tcp","port":"2546","ip":"127.0.0.1"}
{"header":{"cluster_id":"14841639068965178418","member_id":"10276657743932975437","revision":"90","raft_term":"8"}}
InventoryServer Started and ready to accept requests on port 2546
Updating inventory stock on secondary, on Primary's command
Voting to commit the transaction /2214/txp_af53a102-2e3b-45d4-b4c1-3b979e197f9f
Update Type: NEW
totalStock 100
itemId itemStock: 2214 100
Inventory 2214 updated to value 100 committed
Updating inventory stock on secondary, on Primary's command
Voting to commit the transaction /2214/txp_1257baba-f37c-4366-82b0-625e8519f445
Update Type: ORDER
remainingStock 50
itemIdForOrder itemStockForOrder: 2214 50
Inventory 2214 updated to value 50 committed
Updating inventory stock on secondary, on Primary's command
Voting to commit the transaction /2214/txp_813b65e8-241b-4ea6-8852-6fb8b0a1a473
Update Type: NEW
totalStock 100
itemId itemStock: 2214 100
Inventory 2214 updated to value 100 committed
Contesting to acquire lock checkInv
Successful connected to the server
Child node created at /checkInv/lp_0000000078
lock: iit.inv.sync.lock.client.DistributedLock@4238ddc0
I Got the lock at 10:48:55
Request for inventory stock check received..
Checking stock for inventory 2214
Responding, stock for inventory 2214 is 100
Accessing critical section. Time remaining : 5 seconds.
Releasing the lock 10:49:00
Calling Primary server
Call Server localhost:3991
Updating inventory stock on secondary, on Primary's command
Voting to commit the transaction /2214/txp_0ead825d-bc20-4785-92d0-fa38be8b34a1
Update Type: ORDER
remainingStock 50
itemIdForOrder itemStockForOrder: 2214 50
Inventory 2214 updated to value 50 committed

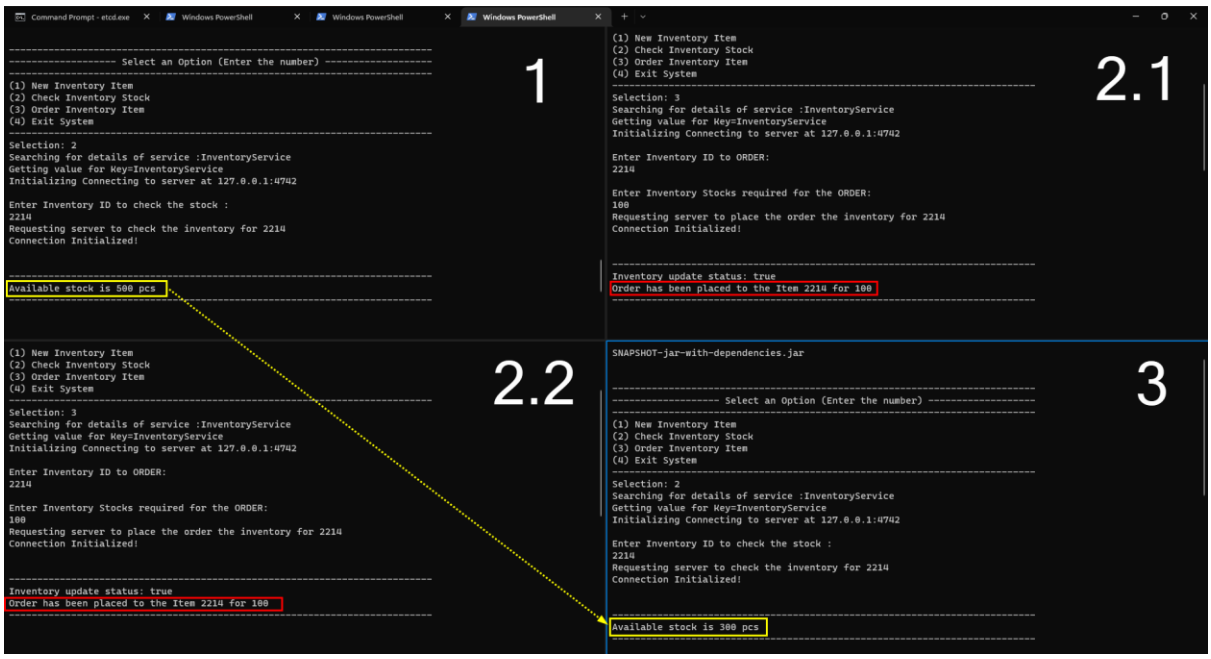


- Below images shows that in a situation where the primary server or any node get shut down, if it's a primary server, the leadership will be transferred to immediately available server.
- And all the new clients will be connected to the next available server.
- And if a new node joins, it'll also be available to the users with updated data. If and when a client request comes to new node, it hands over to leader node to process. Then leader node processes the request and update all other nodes with updated data including the new nodes.





- As per above images, workshop managers can place orders to the system. System will be checking whether the inventory stock is sufficient to place the firstly received order for specific item and if so, the firstly received order for that item will be processed.
- If the system receives another order for the same item at the same time, it'll be processed next by checking the same conditions above and process if the stock is sufficient for the second and/or remaining orders.
    - Below image shows when two orders are placed parallelly AND stock **is sufficient**, orders are processed first-come first-served basis and finalize the transactions.



- Below image shows when two orders are placed parallelly AND stock **is not sufficient**, orders are processed first-come first-served basis and finalize the transactions and the second order is not placed due to insufficient stocks.