Orchestration

Docker Swarm Configuration for InkBlink

Requirements

Tailscale Download (MAC/PC)	Download | Tailscale
Docker Swarm (PC)	Windows | Docker Docs

Initializing Docker Swarm for Inkblink

All machines that run Inkblink must be in the same Tailscale network.

> **WINDOWS** : If docker swarm cannot be initialized using the Tailscale IP address, run a separate Tailscale image using docker. *See **Run Tailscale Using Docker***

```
# Print Tailscale IP
tailscale ip -4
# if running Tailscale on a separate image
# When Tailscale is running:
docker exec -it <tailscale-container-name> tailscale ip -4
```

Once you have found out what the Tailscale IP address for your machine is, initialize a docker swarm using that address

```
# Initialize Docker Swarm
docker swarm init --advertise-addr <YOUR_TAILSCALE_IP> --listen-addr 0.0.0.0:2377
```

To verify the swarm has been initialized successfully:

```
$ netstat -an | grep 2377
  TCP    [::1]:2377             [::]:0                 LISTENING
```

A swarm has been initialized. For other nodes to join the swarm, each node must copy and paste the swarm token in their respective terminals. The token can be retrieved by the manager of the swarm using :

```
# The swarm manager can view nodes and assign nodes tasks
docker swarm join-token worker|manager
```

Expected Output:

```
$ docker swarm join-token worker
To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-0xuent3dqa4feid2p8tfmtbncsbr7idxq1g682jdiiq9bi71ex3-
boafyko6qug1khac6mi38uc17 <YOUR_TAILSCALE_IP>
```

To view joined nodes to your swarm, input:

```
# Run
docker node ls

# Expected output
$ docker node ls
ID                            HOSTNAME         STATUS    AVAILABILITY   MANAGER STATUS
ENGINE VERSION
y5eyvalhfj1umrewfk1eqiqni *   docker-desktop   Ready     Active         Leader
27.5.1
```

Using the listed IDs, the docker swarm manager can now assign roles to each node.

Assigning Roles to Nodes

To assign roles to a node, the service that you want to assign must have a label in **compose.yaml**. A typical compose.yaml file may look like this:

```
  primary_server:
    image: server
    build:
      context: ./backend
    hostname: primary_server
    ports:
      - "8887:8887"
      - "5001:5001"
```

```
      - "5002:5002"
      - "5003:5003"
    environment:
      TAILSCALE_IP: ${PRIMARY_SERVER_IP}
      KAFKA_BOOTSTRAP_SERVERS: ${KAFKA_IP}:9092
      BACKUP_SERVER_1_IP: ${BACKUP_SERVER_1_IP}
      BACKUP_SERVER_2_IP: ${BACKUP_SERVER_2_IP}
      BACKUP_SERVER_3_IP: ${BACKUP_SERVER_3_IP}
      PRIMARY_SERVER_IP: ${PRIMARY_SERVER_IP}
      COORDINATOR_IP: ${COORDINATOR_IP}
    command: >
      sh -c "/wait-for-backups.sh && mvn exec:java -Dexec.mainClass='com.server.WebServer' -
Dexec.args='8887 5001
${BACKUP_SERVER_1_IP}:6001,${BACKUP_SERVER_2_IP}:7001,${BACKUP_SERVER_3_IP}:4001 true'"
    networks:
      - my_network
    deploy:
      placement:
        constraints:
          - node.labels.role == main_services
        # Under deploy/placement/constraints, you can add a label to the service to assign
later to a node
```

Once you have added labels to each service, you can now update each node's role:

```
# Assigning a role to a node
# Only a swarm manager can assign roles to nodes
docker node update --label-add role=<LABEL_NAME> <NODE_ID>
```

Preliminary steps to initialize the service is now complete.

Building Docker Image for Inkblink

All nodes that will run the service must each have the same source files as the manager. Ensure that the source files are up to date with the manager and build the images using:

```
docker compose build
```

The worker node does not run the images independently. Once the swarm initiates the services, the workers will run the respective images automatically.

Deploying Containers using Docker Swarm

The **compose.yaml** utilizes several variables that are imported from **.env** located in the same path as the compose file. The .env file must be updated before building and deploying. The **.env** file may look something like this:

```
# Replace with corresponding tailscale IPs
KAFKA_IP=100.83.215.115
PRIMARY_SERVER_IP=100.83.215.115
BACKUP_SERVER_1_IP=100.83.215.115
BACKUP_SERVER_2_IP=100.83.215.115
BACKUP_SERVER_3_IP=100.83.215.115
COORDINATOR_IP=100.83.215.115
```

Update the values of these variables manually to each node's respective Tailscale IP address.

> **The compose.yaml file does not automatically import these vars when building! Run the following to create a new compose.yaml file**

```
# Export vars in .env to local terminal
export $(grep -v '^#' .env | xargs)

# Create another compose file that manager will use to deploy
envsubst < compose.yaml > rendered-compose.yaml
```

The created **rendered_compose.yaml** file will be used to deploy the containers to the swarm.

The swarm manager can now build the containers using:

```
# Build using new compose file
docker compose -f rendered_compose.yaml build
```

To now deploy the service:

```
# Deploy containers to swarm
docker stack deploy -c rendered-compose.yaml inkblink
```

Confirm that services are running using either Docker Desktop or the terminal:

```
$ docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED
```

```
STATUS                      PORTS      NAMES
4a2130cb26fb   client:latest                  "docker-entrypoint.s…"   About an hour ago   Up
About an hour                         inkblink_client_1.1.u9sic4vg0y0ecuynpykhvnvx9
4246003ed693   server:latest                  "mvn exec:java -Dexe…"   About an hour ago   Up
About an hour            8887/tcp   inkblink_backup_server_3.1.j7orcp0g7bgnzowew2d2pudm6
5a37d51f4152   server:latest                  "mvn exec:java -Dexe…"   About an hour ago   Up
About an hour            8887/tcp   inkblink_backup_server_2.1.v3oyyx2phblj1v1rnnkebvi3f
9de19e542fd2   server:latest                  "mvn exec:java -Dexe…"   About an hour ago   Up
About an hour            8887/tcp   inkblink_backup_server_1.1.qg62zpihy890nqlnl0mkmh73o
28eadabc00e6   server:latest                  "sh -c '/wait-for-ba…"   About an hour ago   Up
About an hour (Paused)   8887/tcp   inkblink_primary_server.1.wyla6fgxpjhovzbriygfefs0v
af844889dc73   connection_coordinator:latest   "mvn exec:java -Dexe…"   About an hour ago   Up
About an hour            8887/tcp
inkblink_connection_coordinator.1.y8voelddpfqjgll2t7a0koyrt
97285295c564   apache/kafka:latest            "/__cacert_entrypoin…"   About an hour ago   Up
About an hour            9092/tcp   inkblink_kafka.1.vv48duzjxu7cx6snpcxperut5
4029d93bd800   tailscale/tailscale            "tailscaled"             23 hours ago
Up 2 hours                            tailscale-proxy
```

WINDOWS : Running Tailscale as a Docker Instance

Create a new compose file, **tailscale-compose.yaml** with the following contents

```
services:
  tailscale-proxy:
    container_name: tailscale-proxy
    image: tailscale/tailscale
    network_mode: "host"
    cap_add:
      - NET_ADMIN
      - SYS_MODULE
    environment:
      - TS_STATE_DIR=/var/lib/tailscale
      - TS_EXTRA_ARGS=--hostname=${HOSTNAME:-proxy-node}
    volumes:
      - tailscale-state:/var/lib/tailscale
      - /dev/net/tun:/dev/net/tun
    restart: unless-stopped
    command: tailscaled
```

```
volumes:
  tailscale-state:
```

Run the following

```
docker compose -f tailscale-compose.yaml build
# Run the tailscale image in the background
docker compose up -d
```

The image above will be running without being previously authenticated to join a Tailscale network instance. To join a network, you must be logged in. To do so, run :

```
# Activate tailscale
# This runs the command `tailscale up` in the container tailscale-proxy
docker exec -it tailscale-proxy tailscale up
```

You'll be prompted to a link which you can copy and paste in your local browser to log in.

RESET EVERYTHING

```
docker system prune -a --volumes
docker image prune --all --force
```

```
docker swarm join --token SWMTKN-1-30aoyw0mju91ytbkezdcatwmg49oyp5gms1btc65ckqzd2v551-
5qlx2du8is6r0jsf88vsxuyh9 100.83.215.115:2377
```

NODOCKER Environment for Inkblink

Requirements

Kafka (Apache)	Kafka Binaries Download

Install Kafka Using Homebrew

```
# Install Homebrew (if you already have homebrew, skip this step
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"

# Install Kafka
brew update
brew install kafka
```

Create kraft.properties File

Create a file *kraft.properties* and place it in the directory homebrew installed Kafka; it may look something like this

```
/usr/local/Cellar/kafka/<version>/libexec
```

<kraft.properties>

```
node.id=1
process.roles=broker,controller
listeners=PLAINTEXT://0.0.0.0:9092,CONTROLLER://127.0.0.1:19092
advertised.listeners=PLAINTEXT://100.78.239.70:9092
controller.quorum.voters=1@127.0.0.1:19092
controller.listener.names=CONTROLLER
listener.security.protocol.map=CONTROLLER:PLAINTEXT,PLAINTEXT:PLAINTEXT
```

```
inter.broker.listener.name=PLAINTEXT
offsets.topic.replication.factor=1
group.initial.rebalance.delay.ms=0
log.retention.hours=1
log.cleanup.policy=delete
message.max.bytes=33554432
```

Format Storage (Using Created Properties File)

```
kafka-storage format -t $(kafka-storage random-uuid) -c /path/to/kraft.properties
```

Start Kafka Server

```
kafka-server-start /path/to/kraft.properties
```

Install Kafka Manually

Extract downloaded tar file

```
tar -xzf kafka_2.13-3.5.1.tgz
cd kafka_2.13-3.5.1
```

Create kraft.properties File

Create a file *kraft.properties* and place it in the extracted directory

```
node.id=1
process.roles=broker,controller
listeners=PLAINTEXT://0.0.0.0:9092,CONTROLLER://127.0.0.1:19092
advertised.listeners=PLAINTEXT://100.78.239.70:9092
controller.quorum.voters=1@127.0.0.1:19092
controller.listener.names=CONTROLLER
listener.security.protocol.map=CONTROLLER:PLAINTEXT,PLAINTEXT:PLAINTEXT
inter.broker.listener.name=PLAINTEXT
offsets.topic.replication.factor=1
group.initial.rebalance.delay.ms=0
log.retention.hours=1
log.cleanup.policy=delete
```

```
message.max.bytes=33554432
```

Format Storage (Using Created Properties File)

```
#In your extracted directory, run
bin/kafka-storage.sh format -t $(bin/kafka-storage.sh random-uuid) -c config/kraft.properties
```

Start Kafka Server

```
bin/kafka-server-start.sh config/kraft.properties
```

Check if Kafka is Running Properly

```
nc -vz <your_ip> 9092
# Or
kafka-topics --bootstrap-server <your_ip>:9092 --list
```

Configure .env for One Machine

```
# Replace with corresponding tailscale IPs
# TAILSCALE_IP is your machine IP
# Replace all others with your TAILSCALE_IP
TAILSCALE_IP=100.78.239.70
KAFKA_IP=100.78.239.70
COORDINATOR_IP=100.76.248.111
PRIMARY_SERVER_IP=100.78.239.70
BACKUP_SERVER_1_IP=100.78.239.70
BACKUP_SERVER_2_IP=100.72.227.86
BACKUP_SERVER_3_IP=100.78.239.70
BACKUP_SERVER_4_IP=100.72.227.86
KAFKA_BOOTSTRAP_SERVERS=100.78.239.70:9092
```

Run Servers Individually

You must run in the following order:

Kafka -> Coordinator -> Backups -> Primary

Have open 7 terminals for each server (including Kafka)

```
# Run export once on each terminal
export $(grep -v '^#' .env | xargs)

# arguments: coordinator, primary, backup1, backup2, backup3, backup4
sh run-server.sh <server>
```

> Restart Kafka (Hard Reset) when you restart servers

```
# for MAC
rm -rf kafka-logs # Run inside Kafka directory

# for Windows
rm -rf C:\tmp\kafka-logs  # Using non PowerShell
Remove-Item -Recurse -Force C:\tmp\kafka-logs # Using PowerShell
```