Introduction to Zookeeper

Taegeon Um

Most contents are from http://www.slideshare.net/sauravhaloi/introduction-to-apache-zookeeper

What is a Distributed System?

A distributed system consists of multiple computers that communicate through a computer network and interact with each other to achieve a common goal.

- Wikipedia

Coordination in a distributed system

- *Coordination*: An act that multiple nodes must perform together.
- Examples:
  - **Group membership**
  - **Locking**
  - **Publisher/Subscriber**
  - **Leader Election**
  - **Synchronization**
- Getting node coordination correct is very hard!

Introducing ZooKeeper

- ZooKeeper allows **distributed processes** to **coordinate** with each other **through a shared hierarchical name space** of data registers.

- A service for **coordinating** processes of distributed applications

- **Provide a simple and high performance kernel** for building more complex coordination primitives at the client.

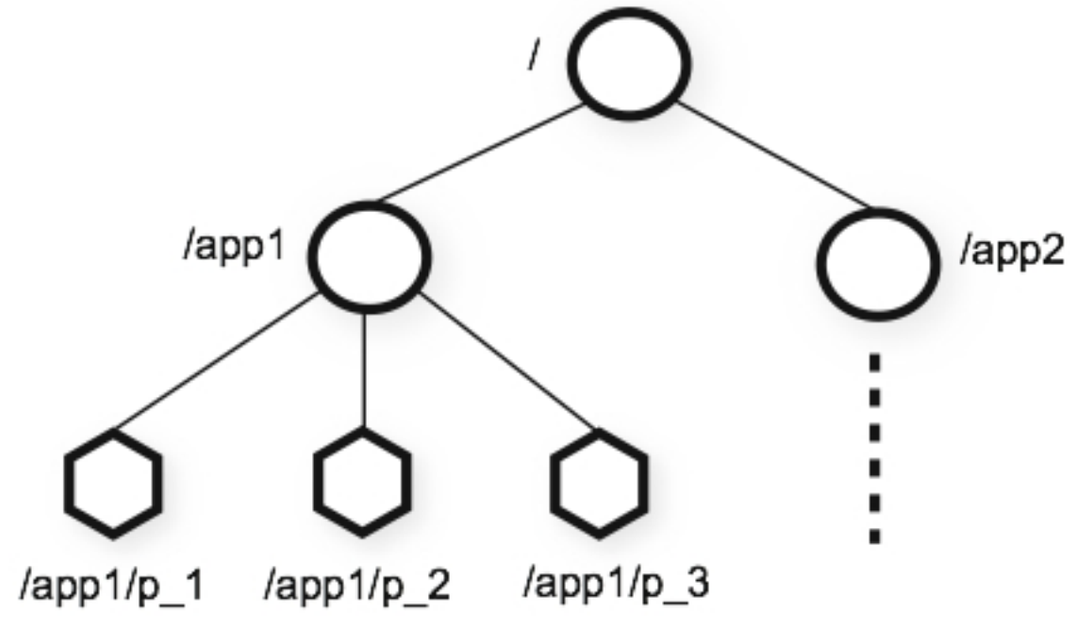
- Wait-free coordination

ZooKeeper Use Cases

- **Configuration Management**
  - Cluster member nodes bootstrapping configuration from a centralized source in unattended way
  - Easier, simpler deployment/provisioning
- **Distributed Cluster Management**
  - Node join / leave
  - Node statuses in real time
- **Naming service – e.g. DNS**
- **Distributed synchronization - locks, barriers, queues**
- **Leader election in a distributed system.**
- **Centralized and highly reliable (simple) data registry**

The ZooKeeper Data Model

- ZooKeeper has a **hierarchal** name space.
- Each node in the namespace is called as a ***ZNode***.
- **Every ZNode has data** (given as byte[]) and can optionally have children.
- Clients can set **watches** on znodes. Changes to that znode trigger the watch and then clear the watch. When a watch triggers, ZooKeeper sends the client a notification.



Znode Types

- Persistent Nodes
  - exists till explicitly deleted
- Ephemeral Nodes
  - exists as long as the session is active
  - can’t have children

- Sequence Nodes (Unique Naming)
  - append a monotonically increasing counter to the end of path
  - applies to both persistent & ephemeral nodes

ZooKeeper API

String create(path, data, acl, createMode)
void create(path, data, acl, createMode, callback, ctx) — **write**

void delete(path, expectedVersion)
void delete(path, expectedVersion, callback, ctx) — **write**

Stat setData(path, data, expectedVersion)
void setData(path, data, expectedVersion, callback, ctx) — **write**

(data, Stat) getData(path, *watch*)
void getData(path, *watch, callback, ctx) — **read**

Stat exists(path, *watch*)
void exists(path, *watch, callback, ctx*) — **read**

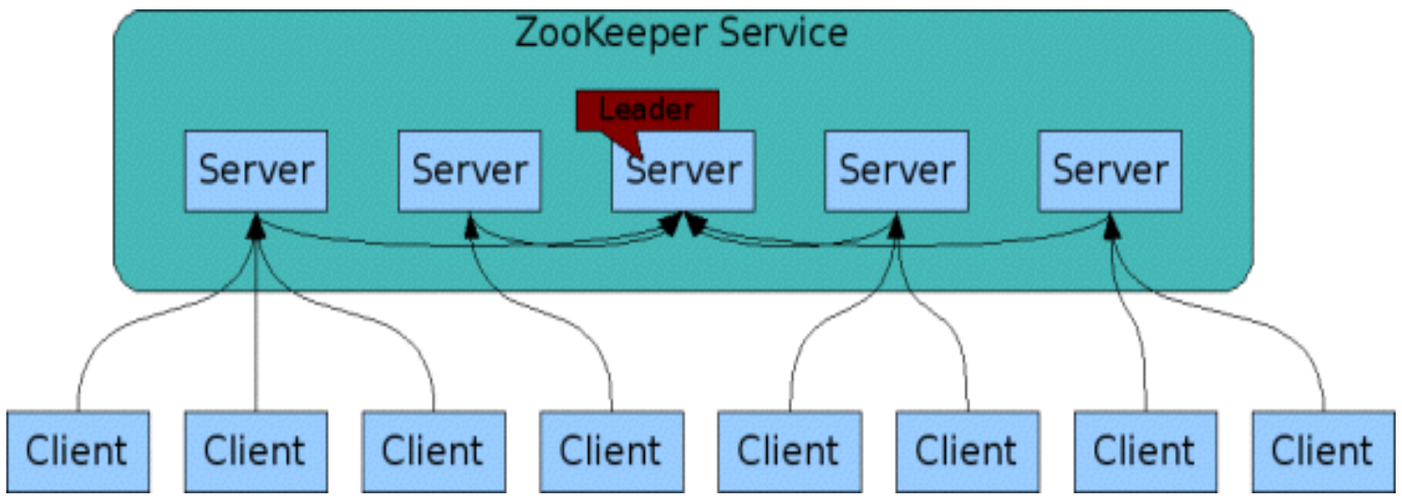
String[] getChildren(path, *watch*)
void getChildren(path, *watch, callback, ctx*) — **read**

void sync(path)

ZooKeeper Watches

- Clients can set watches on znodes:
  - NodeChildrenChanged
  - NodeCreated
  - NodeDataChanged
  - NodeDeleted
- Changes to a znode trigger the watch and ZooKeeper sends the client a notification.
- Watches are **one time triggers**.
- Watches are always **ordered**.
- Client **sees watched event before new znode data**.
- Client should handle cases of latency between getting the event and sending a new request to get a watch.

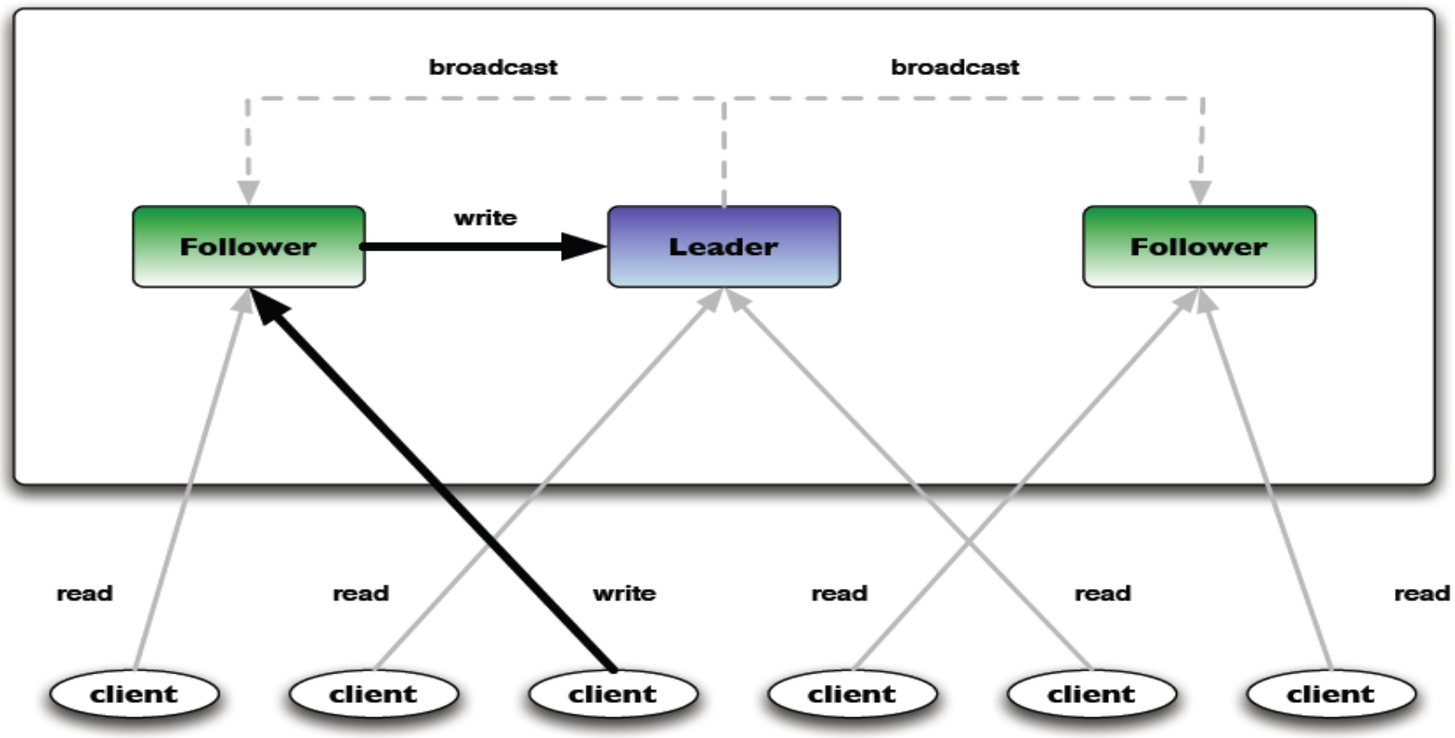
The ZooKeeper Service



- ZooKeeper Service is **replicated** over a set of machines
- All machines **store a copy of the data (in memory)**
- A leader is elected on service startup
- Clients only connect to a single ZooKeeper server & maintains a TCP connection.

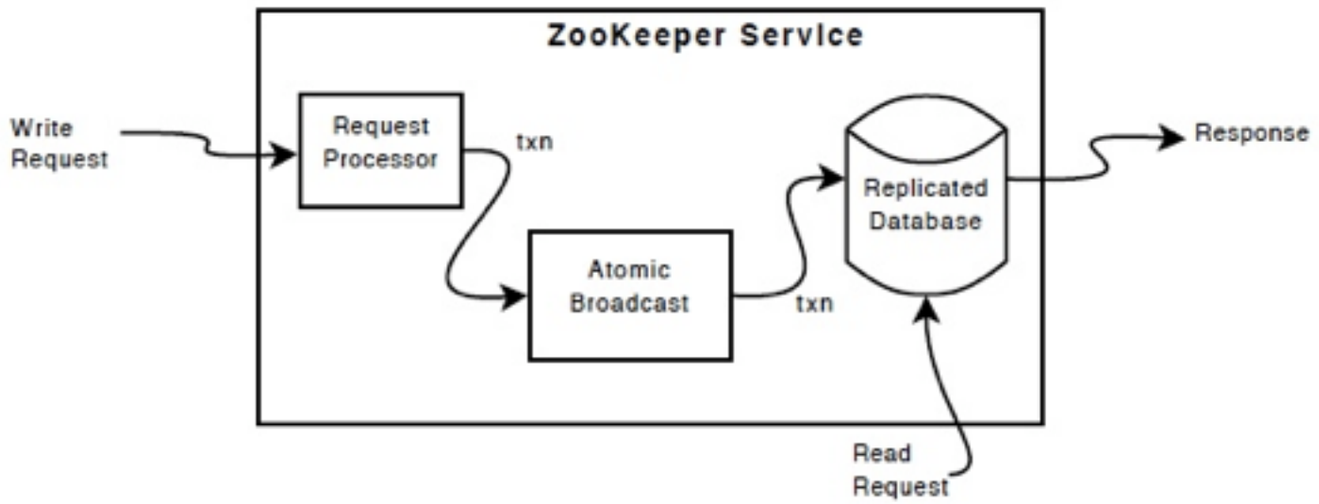
Reads and Writes

Zookeeper Atomic Broadcast protocol



- **Read** requests are processed **locally** at the ZooKeeper server to which the client is currently connected.
- **Write** requests are **forwarded to the leader** and go through majority consensus before a response is generated.
- Designed for workloads with ratios of read to write operations that are higher than 2:1
- Guarantee FIFO client order

Zab: Zookeeper Atomic Broacast protocol



Crash-recovery atomic broadcast algorithm

1. Zab: High-performance broadcast for primary-backup systems
2. A simple totally ordered broadcast protocol

Consistency Guarantees

- **Sequential Consistency**: Updates are applied in order
- **Atomicity**: Updates either succeed or fail
- **Single System Image**: A client sees the same view of the service regardless of the ZK server it connects to.
- **Reliability**: Updates persists once applied, till overwritten by some clients.
- **Timeliness**: The clients' view of the system is guaranteed to be up-to-date within a certain time bound. (Eventual Consistency)

Zookeeper tutorial

Setup and deploy a ZooKeeper in standalone mode

1. Download and install JDK>=1.6, if not already installed. This is required because ZooKeeper server runs on JVM.

2. Download zookeeper from http://apache.mirror.cdnetworks.com/zookeeper/zookeeper-3.4.6/zookeeper-3.4.6.tar.gz

3. untar zookeeper-3.4.6.tar.gz

4. Set up the configuration

    1. cp zookeeper-3.4.6/conf/zoo_sample.cfg zookeeper-3.4.6/conf/zoo.cfg

    2. vim zoo.cfg

5. bin/zkServer.sh start # start zookeeper server

6. bin/zkCli.sh # connect to the server

http://zookeeper.apache.org/doc/r3.1.2/zookeeperStarted.html

Data model practice

1. ls /
2. create /myzookeepernode ‘first_version’ # create a znode
3. get /myzookeepernode # get the zknode info
4. set /myzookeepernode ‘second_version’ # change the znode data
5. create -s /myzookeepernode/mysequential- ‘im_sequential’ # create a sequential node
6. create -s /myzookeepernode/mysequential- ‘also_sequential’
7. get /myzookeepernode/mysequential0000000001
8. delete /myzookeepernode/mysequential0000000000
9. delete /myzookeepernode/mysequential0000000001

Group membership practice

How to implement group membership using Zookeeper?

1. create /mygroup ‘top_node’

2. open another terminal

    1. zkCli.sh # and connect to the server

    2. create -e /mygroup/servergreen ‘iam_servergreen’ # create a ephemeral znode

    3. open another termal and zkCli.sh

    4. create -e /mygroup/serverblue ‘iam_serverblue’

    5. close a terminal and ls /mygroup in zkCli

Configuration management practice

1. create /myconfig ‘setting_1’
2. get /myconfig
3. open a new terminal
    1. zkCli.sh
    2. get /myconfig true # set watch
4. set /myconfig ‘setting_2’ # trigger watcher

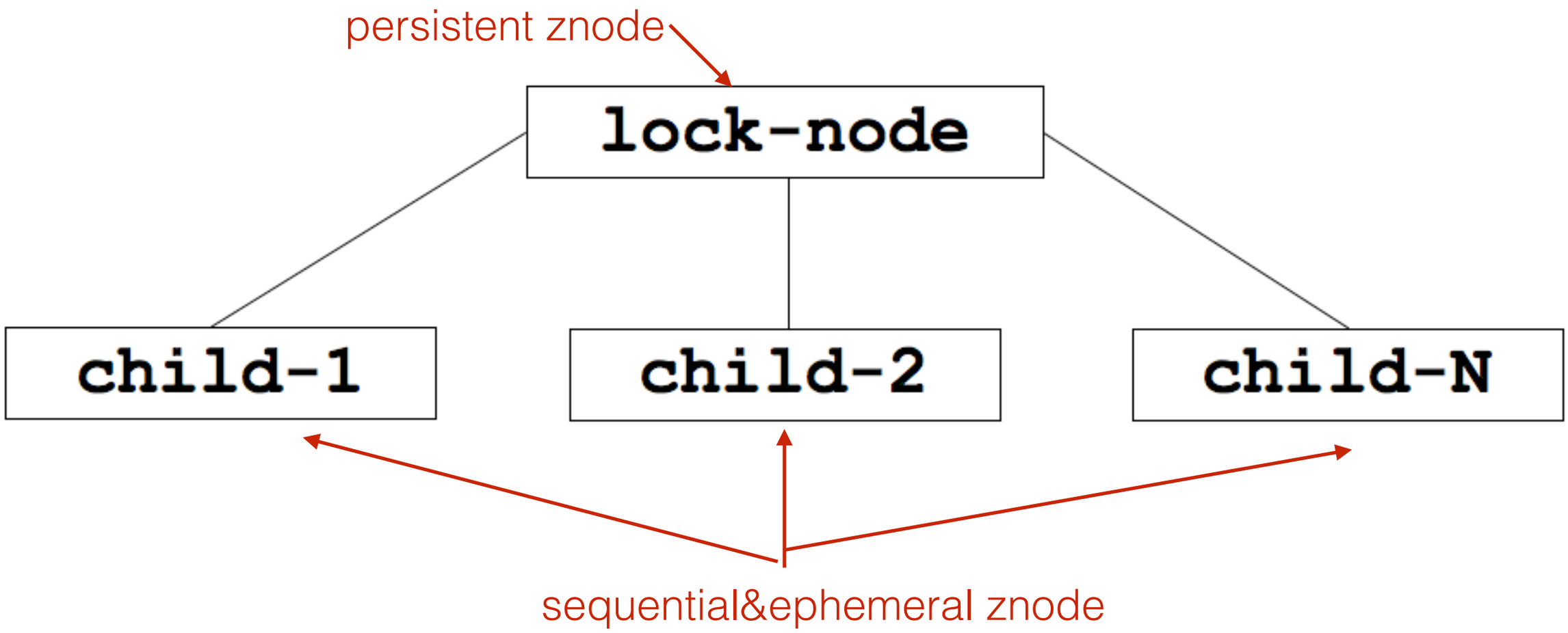
ZooKeeper Java Example

Import example

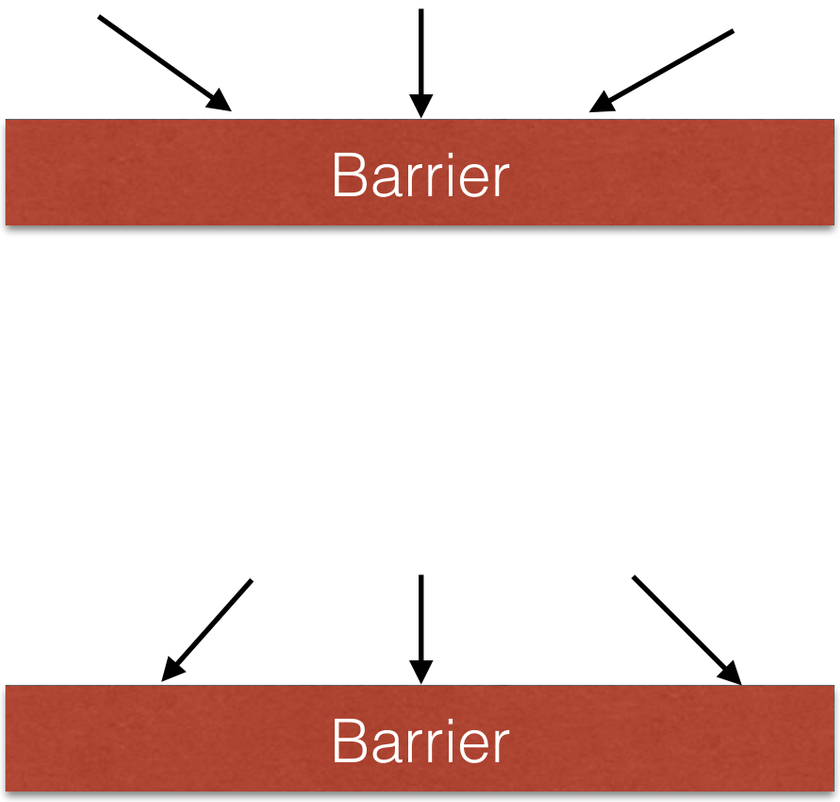
1. git clone https://github.com/taegeonum/zookeeper-exercise.git

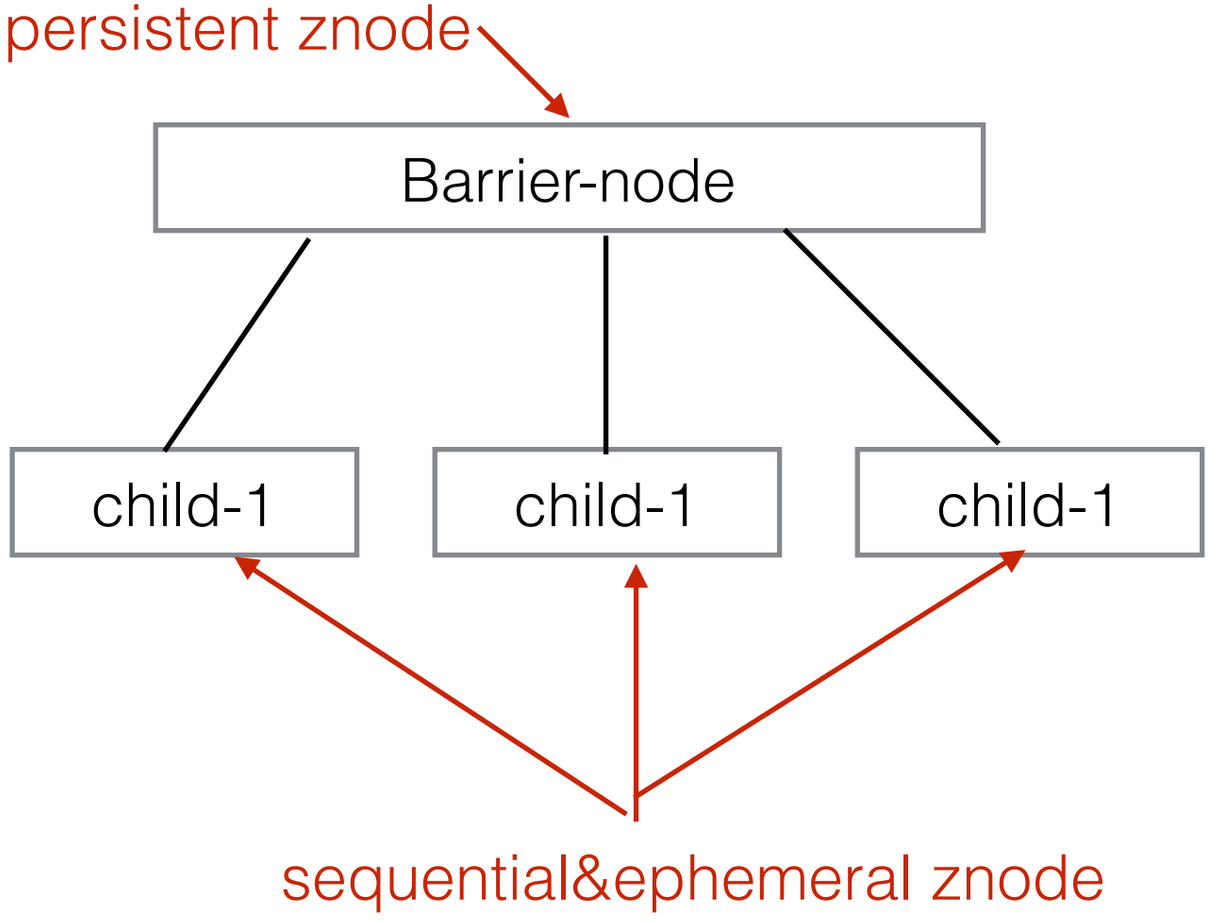
2. Eclipse
    - mvn eclipse:eclipse
    - import > existing projects into workspace
3. Intellij
    - import

Leader election / Distributed Lock

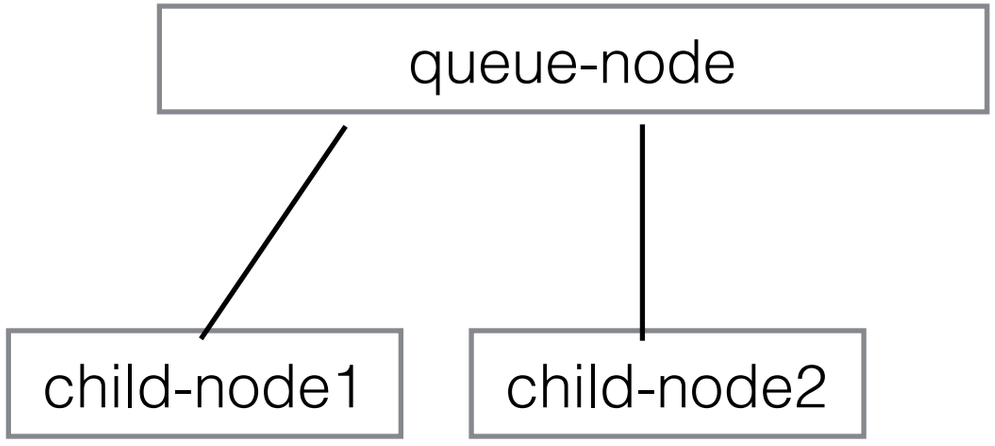


Double Barrier





A Producer/Consumer



- Producer adds persistent_sequential node to queue_node with data
- Consumer retrieves the data and removes the node