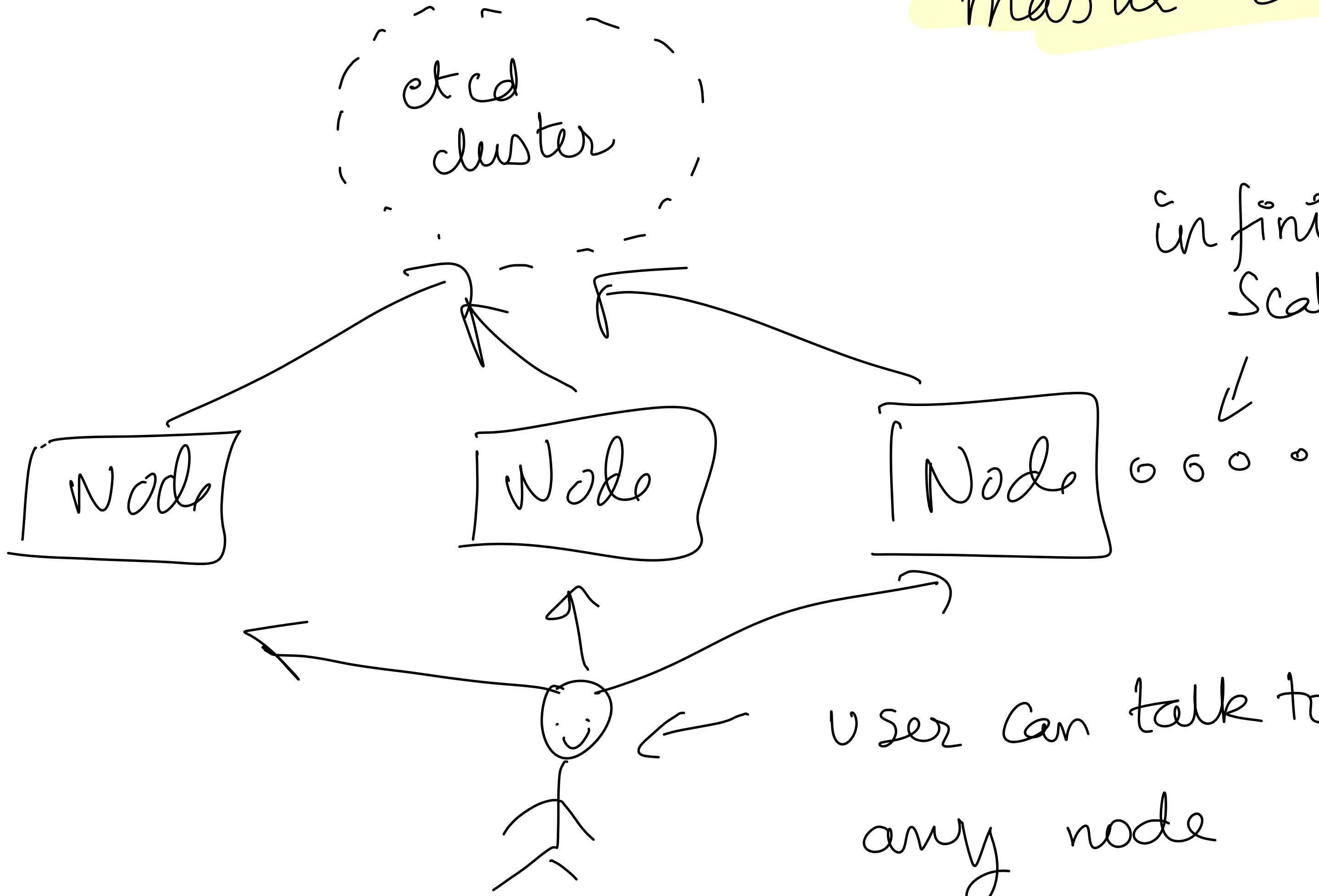
Simple CI

- ☆ Single binary CI System
- ☆ No run modes - every node is both server & worker
- ☆ minimal configuration required

Architecture

~~master - slave~~



user can talk to any node

Cluster Management

☆ Each cluster has a unique-id

☆ ID is a unique string/path

☆ When starting a node, id needs to be provided

- if id is unique, a new cluster is formed
- if id already exists, node joins cluster by that id.

Node Addition

- ☆ when a new node is added, an entry is made into $ID/members/$NODE_ID

  eg:- $ID/members/172.17.31.43

- ☆ Each entry has a TTL of 30 sec

- ☆ If a node doesn't update its entry in 30 sec, it is considered "departed".

Node departure

☆ If a node departs while performing a task, then :-

- Task lock will expire
- Free node will acquire task

☆ If a node departs while free, nothing needs to be done

Work Scheduling

- ☆ When a user request a new task, any node can receive it

- ☆ The node then creates a new entry with the task.json stored in $ID/tasks/$TASK_ID

  eg:- $ID/tasks/abcdefgh

- ☆ Free nodes compete to acquire the task by creating a lock at $ID/tasks/$TASK_ID.lock

Work Scheduling II

- ☆ A lock has a TTL of 30 sec. If not renewed by then, the task is free to be acquired

- ☆ lock holder is responsible for constantly renewing the lock

- ☆ If a task is completed, it is moved to
  $ID/completed/$TASK_ID

Artifact Management

* Push logs & build outputs to S3/minio

System properties

- ☆ Nodes have no awareness of its Siblings
- ☆ Truly follows micro service architecture
- ☆ Idempotent task executions
- ☆ Stateless nodes
- ☆ As resilient as etcd

Kubernetes Operations

- Ideal environment
- Create a deployment with one pod running simple-ci
- Add a Horizontal Pod Autoscaler to build your own elastic CI service

Unknowns

* log streaming
* authentication
* Integrations