

---

## The TinyKV Course

The TinyKV course builds a key-value storage system with the Raft consensus algorithm. It is inspired by MIT 6.824 and TiKV Project.

After completing this course, you will have the knowledge to implement a horizontally scalable, highly available, key-value storage service with distributed transaction support. Also, you will have a better understanding of TiKV architecture and implementation.

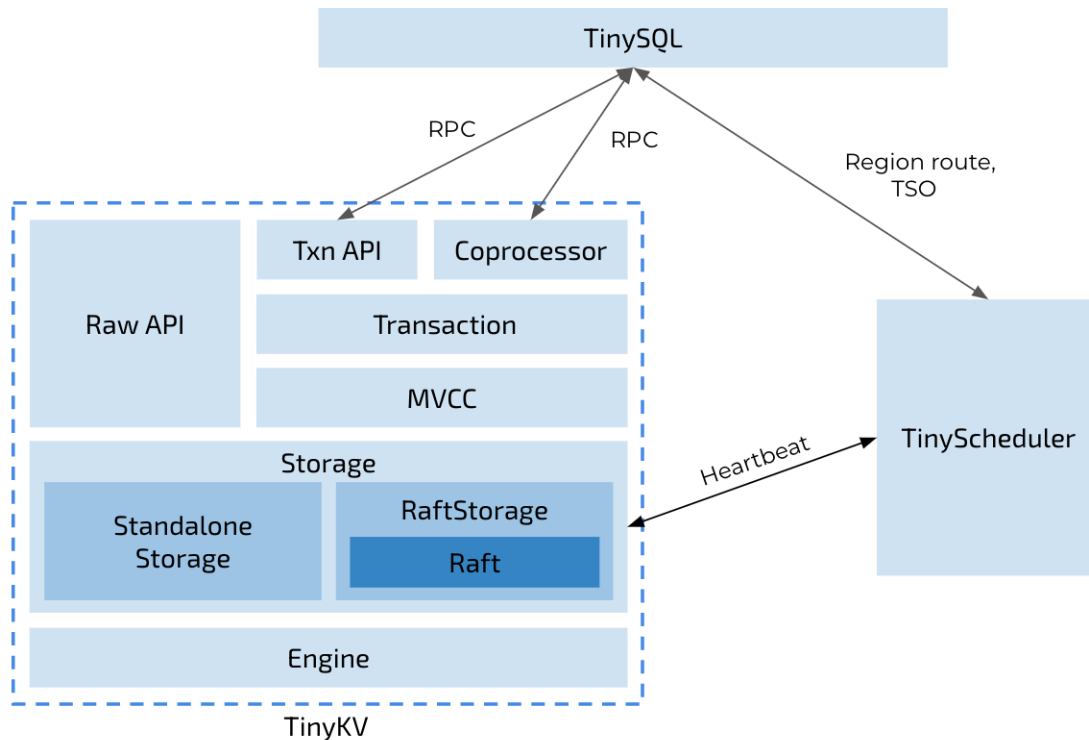
### Course Architecture

The whole project is a skeleton code for a key-value server and a scheduler server at the beginning - you need to finish the core logic step by step:

- Standalone KV
  - Implement a standalone storage engine.
  - Implement raw key-value service handlers.
- Raft KV
  - Implement the basic Raft algorithm.
  - Build a fault-tolerant KV server on top of Raft.
  - Add the support of Raft log garbage collection and snapshot.
- Multi-raft KV
  - Implement membership change and leadership change to Raft algorithm.
  - Implement conf change and region split on Raft store.
  - Implement a basic scheduler.
- Transaction
  - Implement the multi-version concurrency control layer.
  - Implement handlers of `KvGet`, `KvPrewrite`, and `KvCommit` requests.
  - Implement handlers of `KvScan`, `KvCheckTxnStatus`, `KvBatchRollback`, and `KvResolveLock` requests.

---

## Code Structure



Similar to the architecture of TiDB + TiKV + PD that separates the storage and computation, TinyKV only focuses on the storage layer of a distributed database system. If you are also interested in the SQL layer, please see TinySQL. Besides that, there is a component called TinyScheduler acting as a center control of the whole TinyKV cluster, which collects information from the heartbeats of TinyKV. After that, the TinyScheduler can generate scheduling tasks and distribute the tasks to the TinyKV instances. All of instances are communicated via RPC.

The whole project is organized into the following directories:

- `kv` contains the implementation of the key-value store.
- `raft` contains the implementation of the Raft consensus algorithm.
- `scheduler` contains the implementation of the TinyScheduler, which is responsible for managing TinyKV nodes and generating timestamps.
- `proto` contains the implementation of all communication between nodes and processes uses Protocol Buffers over gRPC. This package contains the protocol definitions used by TinyKV, and the generated Go code that you can use.
- `log` contains utility to output log based on level.

---

## Reading List

We provide a reading list for the knowledge of distributed storage system. Though not all of them are highly related with this course, they can help you construct the knowledge system in this field.

Also, you're encouraged to read the overview of TiKV's and PD's design to get a general impression on what you will build:

- TiKV, the design of data storage (English, Chinese).
- PD, the design of scheduling (English, Chinese).

## Build TinyKV from Source

### Prerequisites

- [git](#): The source code of TinyKV is hosted on GitHub as a git repository. To work with git repository, please install [git](#).
- [go](#): TinyKV is a Go project. To build TinyKV from source, please install [go](#) with version greater or equal to 1.13.

### Clone

Clone the source code to your development machine.

```
1 git clone https://github.com/tidb-incubator/tinykv.git
```

### Build

Build TinyKV from the source code.

```
1 cd tinykv
2 make
```

It builds the binary of `tinykv-server` and `tinyscheduler-server` to `bin` dir.

## Run TinyKV with TinySQL

1. Get `tinysql-server` follow its document.
2. Put the binary of `tinyscheduler-server`, `tinykv-server` and `tinysql-server` into a single dir.

---

3. Under the binary dir, run the following commands:

```
1 mkdir -p data
2 ./tinyscheduler-server
3 ./tinykv-server -path=data
4 ./tinysql-server --store=tikv --path="127.0.0.1:2379"
```

Now you can connect to the database with an official MySQL client:

```
1 mysql -u root -h 127.0.0.1 -P 4000
```

## Autograding and certification

Since Jun 2022, we start using github classroom to accept labs and provide autograding timely. The github classroom invitation is <https://classroom.github.com/a/cdINNrFU>. The discussion Wechat/S-lack group and the certification after you pass the class is provided in the tinyKV learning class

Autograding is a workflow which can automatically run test cases and give feedback timely. However there are some limitations in Github classroom, in order to make golang work and run it in our self-hosted machines, **you need to overwrite the workflow generated by Github classroom and commit it.**

```
1 cp scripts/classroom.yml .github/workflows/classroom.yml
2 git add .github
3 git commit -m"update github classroom workflow"
```

## Contributing

Any feedback and contribution is greatly appreciated. Please see issues if you want to join in the development.