[](https://ida.interchain.io/)

[Interchain Developer Academy](https://ida.interchain.io/)/[Interchain Developer Academy](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)



Search

[Interchain Developer Academy](https://ida.interchain.io/)[Interchain Developer Academy](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

Search



Filters

Interchain Developer Academy

[](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Week 0 - Getting Started](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Getting Started](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Blockchain 101](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Blockchain History](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Public and Managed Blockchains](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Consensus in Distributed Networks](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Cryptography](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Self-Assessment Quiz](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Go Introduction - First Steps](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Go Basics](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Go Interfaces](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Control Structures in Go](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Arrays and Slices in Go](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Standard Packages in Go](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Concurrency in Go](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Good-To-Know Dev Terms](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Docker Introduction](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Week 1 - Introduction to the Interchain](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Introduction to the Interchain](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Blockchain Technology and the Interchain](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[The Interchain Ecosystem](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Getting ATOM and Staking It](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[A Blockchain App Architecture](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Accounts](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Transactions](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Messages](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Modules](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Protobuf](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Multistore and Keepers](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[BaseApp](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Queries](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Events](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Context](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Testing](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Relaying with IBC](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Interchain Security](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Bridges](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Migrations](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Week 1 Quiz](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Week 2 - First Steps](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[First Steps](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Setup Your Work Environment](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Run a Node, API, and CLI](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Ignite CLI](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Exercise - Make a Checkers Blockchain](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Store Object](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Create Custom Messages](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Create and Save a Game Properly](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Add a Way to Make a Move](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Emit Game Information](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Record the Game Winner](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Week 2 Exercise](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Week 3 - Introduction to IBC and CosmJS](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Introduction to IBC and CosmJS](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[What is IBC?](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[IBC/TAO - Connections (OPTIONAL)](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[IBC/TAO - Channels (OPTIONAL)](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[IBC/TAO - Clients (OPTIONAL)](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[IBC Token Transfer](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Interchain Accounts (OPTIONAL)](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[IBC Middleware (OPTIONAL)](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Create a Custom IBC Middleware (OPTIONAL)](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Integrate IBC Middleware Into a Chain (OPTIONAL)](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[IBC Tooling](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[What is CosmJS?](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Your First CosmJS Actions](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Compose Complex Transactions](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Learn to Integrate Keplr](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Create Custom CosmJS Interfaces](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Week 4 - Ignite CLI and IBC Advanced](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Ignite CLI and IBC Advanced](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Keep an Up-To-Date Game Deadline](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Keep Track Of How Many Moves Have Been Played](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Put Your Games in Order](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Auto-Expiring Games](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Let Players Set a Wager](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Handle wager payments](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Integration tests](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Incentivize Players](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Help Find a Correct Move](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Play With Cross-Chain Tokens](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Understand IBC Denoms](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Go Relayer](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Hermes Relayer](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Week 5 - CosmJS Advanced](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[CosmJS Advanced](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Create Custom Objects](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Create Custom Messages](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Get an External GUI](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Integrate CosmJS and Keplr](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Backend Script for Game Indexing](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Week 6 - IBC Deep Dive](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[IBC Deep Dive](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[IBC Application Developer Introduction](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Make a Module IBC-Enabled](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Adding Packet and Acknowledgment Data](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Extend the Checkers Game With a Leaderboard](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Create a Leaderboard Chain](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Week 7 - From Code to MVP to Production and Migrations](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[From Code to MVP to Production and Migrations](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Run in Production](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Prepare the Software to Run](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Prepare a Validator and Keys](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Prepare Where the Node Starts](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Prepare and Connect to Other Nodes](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Configure, Run, and Set Up a Service](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Prepare and Do Migrations](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Simulate Production in Docker](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Tally Player Info After Production](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Add a Leaderboard as a Module](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Migrate the Leaderboard Module After Production](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Simulate a Migration in Docker](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Final Exam](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[What's Next?](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

[Continue Your Interchain Journey](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html)

Docs Version Switcher

On this page

[Some initial thoughts](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html#some-initial-thoughts)

[Code needs](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html#code-needs)

[Defining the rule set](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html#defining-the-rule-set)

[The stored game object](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html#the-stored-game-object)

[How to store](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html#how-to-store)

[Looking around](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html#looking-around)

[Protobuf objects](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html#protobuf-objects)

[How Ignite CLI works](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html#how-ignite-cli-works)

[Files to adjust](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html#files-to-adjust)

[Protobuf service interfaces](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html#protobuf-service-interfaces)

[Additional helper functions](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html#additional-helper-functions)

[Unit tests](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html#unit-tests)

[Your first unit test](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html#your-first-unit-test)

[Debug your unit test](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html#debug-your-unit-test)

[More unit tests](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html#more-unit-tests)

[Interact via the CLI](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html#interact-via-the-cli)

[#Copy link](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html#store-object) **Store Object**



Make sure you have all you need before proceeding with the exercise:

* You understand the concepts of [accounts](https://ida.interchain.io/academy/2-cosmos-concepts/2-accounts.html), [Protobuf](https://ida.interchain.io/academy/2-cosmos-concepts/6-protobuf.html), and [multistore](https://ida.interchain.io/academy/2-cosmos-concepts/7-multistore-keepers.html).
* Go is installed.
* You have the bare blockchain scaffold codebase with a single module named checkers. If not, follow the [previous steps](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/1-ignitecli.html) or check out the [relevant version (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/tree/ignite-start).
* You have the checkers\_i Docker image if you work with Docker. If not, follow the [previous steps](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/1-ignitecli.html).



In this section, you will handle:

* The Stored Game object
* Protobuf objects
* Query.proto
* Protobuf service interfaces
* Your first unit test
* Interactions via the command-line

In the [Ignite CLI introduction section](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/1-ignitecli.html) you learned how to start a completely new blockchain. Now it is time to dive deeper and explore how you can create a blockchain to play a decentralized game of checkers.

[#Copy link](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html#some-initial-thoughts) Some initial thoughts

As you are face-to-face with the proverbial blank page: where do you start?

A good place to start with is thinking about the objects you keep in storage. **A game**, obviously... but what does any game have to keep in storage?

Questions to ask that could influence your design include, but are not limited to:

* What is the lifecycle of a game?
* How are participants selected to be in a game?
* What fields make it possible to play across a span of time and transactions?
* What fields make it possible to differentiate between different games?
* How do you ensure safety against malice, sabotage, or even simple errors?
* What limitations does your design **intentionally** impose on participants?
* What limitations does your design **unintentionally** impose on participants?

After thinking about what goes into each individual game, you should consider the demands of the wider system. In general terms, before you think about the commands that achieve what you seek, ask:

* How do you lay games in storage?
* How do you save and retrieve games?

The goal here is not to finalize every conceivable game feature immediately. For instance, handling wagers or leaderboards can be left for another time. But there should be a basic game design good enough to accommodate future improvements.

[#Copy link](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html#code-needs) Code needs

**Do not** dive headlong into coding the rules of checkers in Go - examples will already exist which you can put to use. Your job is to make a blockchain that just happens to enable the game of checkers.

With that in mind:

* What Ignite CLI commands will get you a long way when it comes to implementation?
* How do you adjust what Ignite CLI created for you?
* How would you unit-test your modest additions?
* How would you use Ignite CLI to locally run a one-node blockchain and interact with it via the CLI to see what you get?

Run the commands, make the adjustments, and run some tests regarding game storage. Do not go into deeper issues like messages and transactions yet.

[#Copy link](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html#defining-the-rule-set) Defining the rule set

A good start to developing a checkers blockchain is to define the rule set of the game. There are many versions of the rules. Choose [a very simple set of basic rules (opens new window)↗](https://www.ducksters.com/games/checkers_rules.php) to avoid getting lost in the rules of checkers or the proper implementation of the board state.

Use [a ready-made implementation (opens new window)↗](https://github.com/batkinson/checkers-go/blob/a09daeb/checkers/checkers.go) with the additional rule that the board is 8x8, is played on black cells, and black plays first. This code will not need adjustments. Copy this rules file into a rules folder inside your module. Change its package from checkers to rules. You can do this by command-line:



Copy

$ mkdir x/checkers/rules

$ curl https://raw.githubusercontent.com/batkinson/checkers-go/a09daeb1548dd4cc0145d87c8da3ed2ea33a62e3/checkers/checkers.go | sed 's/package checkers/package rules/' > x/checkers/rules/checkers.go

Do not focus on the GUI, this procedure lays the foundation for an interface.

Now it is time to create the first object.

[#Copy link](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html#the-stored-game-object) The stored game object

Begin with the minimum game information needed to be stored:

* **Black player.** A string, the serialized address.
* **Red player.** A string, the serialized address.
* **Board proper.** A string, the board as it is serialized by the *rules* file.
* **Player to play next.** A string, specifying whose *turn* it is.



When you save strings, it makes it easier to understand what comes straight out of storage, but at the expense of storage space. As an advanced consideration, you could store the same information in binary.

[#Copy link](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html#how-to-store) How to store

After you know **what** to store, you have to decide **how** to store a game. This is important if you want your blockchain application to accommodate multiple simultaneous games. The game is identified by a unique ID.

How should you generate the ID? If you let players choose it themselves, this could lead to transactions failing because of an ID clash. You cannot rely on a large random number like a universally unique identifier (UUID), because transactions have to be verifiable in the future. Verifiable means that nodes verifying the block need to arrive at the same conclusion. However, the new UUID() command is not deterministic. In this context, it is better to have a counter incrementing on each new game. This is possible because the code execution happens in a single thread.

The counter must be kept in storage between transactions. Instead of a single counter in storage, you can keep the counter in a unique object at a singular storage location, and easily add relevant elements to the object as needed in the future. Name the counter as nextId and its container as SystemInfo.

As for the game type, you can name it as StoredGame.

You can rely on Ignite CLI's assistance for both the counter and the game:

* For the counter and its container, you instruct Ignite CLI with scaffold single:

**Local**

**Docker**



Copy

$ ignite scaffold single systemInfo nextId:uint \

--module checkers \

--no-message

Copy

$ docker run --rm -it \

-v $(pwd):/checkers \

-w /checkers \

checkers\_i \

ignite scaffold single systemInfo nextId:uint \

--module checkers \

--no-message

In this command:

* + nextId is explicitly made to be a uint. If you left it to Ignite's default, it would be a string.
  + You must add --no-message. If you omit it, Ignite CLI creates an sdk.Msg and an associated service whose purpose is to overwrite your SystemInfo object. However, your SystemInfo.NextId must be controlled/incremented by the application and not by a player sending a value of their own choosing. Ignite CLI still creates convenient getters.
* For the game type, because you are storing games by ID, you need a map. Instruct Ignite CLI with scaffold map using the StoredGame name:

**Local**

**Docker**



Copy

$ ignite scaffold map storedGame board turn black red \

--index index \

--module checkers \

--no-message

Copy

$ docker run --rm -it \

-v $(pwd):/checkers \

-w /checkers \

checkers\_i \

ignite scaffold map storedGame board turn black red \

--index index \

--module checkers \

--no-message

In this command:

* + board, turn, black and red are by default strings, so there is no need to be explicit with for instance board:string.
  + index is the id field picked, and anyway is the default name when scaffolding a map. id cannot be chosen when scaffolding with Ignite.
  + --no-message prevents game objects from being created or overwritten with a simple sdk.Msg. The application instead creates and updates the objects when receiving properly crafted messages like [*create game*](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/4-create-message.html) or [*play a move*](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/6-play-game.html).

The Ignite CLI scaffold command creates several files, as you can see [here (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/commit/d5a93bf) and [here (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/commit/8679295).

[#Copy link](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html#looking-around) Looking around

The command added new constants:



Copy

const (

SystemInfoKey = "SystemInfo-value-"

)

x /

checkers /

types /

keys.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/stored-game/x/checkers/types/keys.go" \l "L24-L26" \t "_blank)



Copy

const (

StoredGameKeyPrefix = "StoredGame/value/"

)

x /

checkers /

types /

key\_stored\_game.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/stored-game/x/checkers/types/key_stored_game.go" \l "L7-L10" \t "_blank)

These constants are used as prefixes for the keys that can access the storage location of objects.

In the case of games, the store model lets you *narrow* the search. For instance:



Copy

store := prefix.NewStore(ctx.KVStore(k.storeKey), types.KeyPrefix(types.StoredGameKeyPrefix))

x /

checkers /

keeper /

stored\_game.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/stored-game/x/checkers/keeper/stored_game.go" \l "L24" \t "_blank)

This gets the store to access any game if you have its index:



Copy

b := store.Get(types.StoredGameKey(

index,

))

x /

checkers /

keeper /

stored\_game.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/stored-game/x/checkers/keeper/stored_game.go" \l "L26-L28" \t "_blank)

[#Copy link](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html#protobuf-objects) Protobuf objects

Ignite CLI creates the Protobuf objects in the proto directory before compiling them. The SystemInfo object looks like this:



Copy

message SystemInfo {

uint64 nextId = 1;

}

proto /

checkers /

system\_info.proto

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/stored-game/proto/checkers/system_info.proto" \l "L6-L8" \t "_blank)

The StoredGame object looks like this:



Copy

message StoredGame {

string index = 1;

string board = 2;

string turn = 3;

string black = 4;

string red = 5;

}

proto /

checkers /

stored\_game.proto

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/stored-game/proto/checkers/stored_game.proto" \l "L6-L12" \t "_blank)

Both objects compile to:



Copy

type SystemInfo struct {

NextId uint64 `protobuf:"varint,1,opt,name=nextId,proto3" json:"nextId,omitempty"`

}

x /

checkers /

types /

system\_info.pb.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/stored-game/x/checkers/types/system_info.pb.go" \l "L25-L27" \t "_blank)

And to:



Copy

type StoredGame struct {

Index string `protobuf:"bytes,1,opt,name=index,proto3" json:"index,omitempty"`

Board string `protobuf:"bytes,2,opt,name=board,proto3" json:"board,omitempty"`

Turn string `protobuf:"bytes,3,opt,name=turn,proto3" json:"turn,omitempty"`

Black string `protobuf:"bytes,4,opt,name=black,proto3" json:"black,omitempty"`

Red string `protobuf:"bytes,5,opt,name=red,proto3" json:"red,omitempty"`

}

x /

checkers /

types /

stored\_game.pb.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/stored-game/x/checkers/types/stored_game.pb.go" \l "L25-L31" \t "_blank)



At this point, note that SystemInfo and StoredGame do not have a field named *creator*. That is because they were created with the --no-message flag. If you had omitted this flag, the message creator would always be saved into the object's creator. Like so:



Copy

message SystemInfo {

string creator = 1; // If you had omitted --no-message

uint64 nextId = 2;

}

These are not the only created Protobuf objects. The genesis state is also defined in Protobuf:



Copy

import "checkers/system\_info.proto";

import "checkers/stored\_game.proto";

message GenesisState {

...

SystemInfo systemInfo = 2;

repeated StoredGame storedGameList = 3 [(gogoproto.nullable) = false];

}

proto /

checkers /

genesis.proto

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/4249ce1/proto/checkers/genesis.proto" \l "L15-L16" \t "_blank)

This is compiled to:



Copy

type GenesisState struct {

Params Params `protobuf:"bytes,1,opt,name=params,proto3" json:"params"`

SystemInfo \*SystemInfo `protobuf:"bytes,2,opt,name=systemInfo,proto3" json:"systemInfo,omitempty"`

StoredGameList []StoredGame `protobuf:"bytes,3,rep,name=storedGameList,proto3" json:"storedGameList"`

}

x /

checkers /

types /

genesis.pb.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/stored-game/x/checkers/types/genesis.pb.go" \l "L26-L30" \t "_blank)

You can find query objects as part of the boilerplate objects created by Ignite CLI. Ignite CLI creates the objects according to a model, but this does not prevent you from making changes later if you decide these queries are not needed:



Copy

message QueryGetSystemInfoRequest {}

message QueryGetSystemInfoResponse {

SystemInfo SystemInfo = 1 [(gogoproto.nullable) = false];

}

proto /

checkers /

query.proto

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/stored-game/proto/checkers/query.proto" \l "L46-L50" \t "_blank)

The query objects for StoredGame are more useful for your checkers game, and look like this:



Copy

message QueryGetStoredGameRequest {

string index = 1;

}

message QueryGetStoredGameResponse {

StoredGame StoredGame = 1 [(gogoproto.nullable) = false];

}

message QueryAllStoredGameRequest {

cosmos.base.query.v1beta1.PageRequest pagination = 1;

}

message QueryAllStoredGameResponse {

repeated StoredGame StoredGame = 1 [(gogoproto.nullable) = false];

cosmos.base.query.v1beta1.PageResponse pagination = 2;

}

proto /

checkers /

query.proto

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/stored-game/proto/checkers/query.proto" \l "L51-L67" \t "_blank)

[#Copy link](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html#how-ignite-cli-works) How Ignite CLI works

Ignite CLI puts the different Protobuf messages into different files depending on their use:

* [**query.proto** (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/blob/stored-game/proto/checkers/query.proto) - for objects related to reading the state. Ignite CLI modifies this file whenever you instruct it to add queries. This includes objects to [query your stored elements (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/blob/stored-game/proto/checkers/query.proto#L46-L67).
* [**tx.proto** (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/blob/stored-game/proto/checkers/tx.proto) - for objects that relate to updating the state. As you have only defined storage elements with --no-message, it is empty for now. The file will be modified as you add transaction-related elements like the message to [create a game](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/4-create-message.html).
* [**genesis.proto** (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/blob/stored-game/proto/checkers/genesis.proto) - for the genesis. Ignite CLI modifies this file according to how your new storage elements evolve.
* [**system\_info.proto** (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/blob/stored-game/proto/checkers/system_info.proto) and [**stored\_game.proto** (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/blob/stored-game/proto/checkers/stored_game.proto) - separate files created once, that will remain untouched by Ignite CLI. You are free to modify them but be careful with [field numbering (opens new window)↗](https://developers.google.com/protocol-buffers/docs/overview#assigning_field_numbers).

Files updated by Ignite CLI include comments like:



Copy

// this line is used by starport scaffolding # 2

proto /

checkers /

query.proto

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/stored-game/proto/checkers/query.proto" \l "L34" \t "_blank)



Ignite CLI adds code right below the comments, which explains why at times the oldest lines appear lower than recent ones. Make sure to keep these comments where they are so that Ignite CLI knows where to inject code in the future. You could add your code above or below the comments.

Some files created by Ignite CLI can be updated, but you should not modify the Protobuf-compiled files [\*.pb.go (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/blob/stored-game/x/checkers/types/system_info.pb.go) and [\*.pb.gw.go (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/blob/stored-game/x/checkers/types/query.pb.gw.go) as they are recreated on every re-run of ignite generate proto-go or equivalent.

[#Copy link](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html#files-to-adjust) Files to adjust

Ignite CLI creates files that you can and should update. For example, the default genesis values start as:



Copy

return &GenesisState{

SystemInfo: nil,

...

}

x /

checkers /

types /

genesis.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/stored-game/x/checkers/types/genesis.go" \l "L13" \t "_blank)

This is not correct. Your chain needs to start with an initial system info. This raises the point that the genesis' SystemInfo should in fact [never be null (opens new window)↗](https://pkg.go.dev/github.com/gogo/protobuf/gogoproto). You can enforce that in genesis.proto:



Copy

message GenesisState {

...

- SystemInfo systemInfo = 2;

+ SystemInfo systemInfo = 2 [(gogoproto.nullable) = false];

...

}

proto /

checkers /

genesis.proto

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/full-game-object/proto/checkers/genesis.proto" \l "L15" \t "_blank)

After compilation, this nullable = false flag changes the SystemInfo type in genesis from a pointer to a straight value. Make sure you recompile:

**Local**

**Docker**



Copy

$ ignite generate proto-go

Copy

$ docker run --rm -it \

-v $(pwd):/checkers \

-w /checkers \

checkers\_i \

ignite generate proto-go

Then set a default value for SystemInfo:



Copy

const DefaultIndex uint64 = 1

func DefaultGenesis() \*GenesisState {

return &GenesisState{

- SystemInfo: nil,

+ SystemInfo: SystemInfo{

+ NextId: uint64(DefaultIndex),

+ },

StoredGameList: []StoredGame{},

...

}

}

x /

checkers /

types /

genesis.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/full-game-object/x/checkers/types/genesis.go" \l "L13-L15" \t "_blank)

You can choose to start with no games or insert a number of games to start with. In either case, you must choose the first ID of the first future created game, which here is set at 1 by reusing the DefaultIndex value.

Do not forget to fix the [other compilation errors (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/commit/58f4adc) due to the change of type.

As you can see, it is possible to adjust what Ignite CLI created.

[#Copy link](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html#protobuf-service-interfaces) Protobuf service interfaces

In addition to created objects, Ignite CLI also creates services that declare and define how to access the newly-created storage objects. Ignite CLI introduces empty service interfaces that can be filled as you add objects and messages when scaffolding a brand new module.

In this case, Ignite CLI added to service Query how to query for your objects:



Copy

service Query {

rpc Params(QueryParamsRequest) returns (QueryParamsResponse) {

option (google.api.http).get = "/alice/checkers/checkers/params";

}

rpc SystemInfo(QueryGetSystemInfoRequest) returns (QueryGetSystemInfoResponse) {

option (google.api.http).get = "/alice/checkers/checkers/system\_info";

}

rpc StoredGame(QueryGetStoredGameRequest) returns (QueryGetStoredGameResponse) {

option (google.api.http).get = "/alice/checkers/checkers/stored\_game/{index}";

}

rpc StoredGameAll(QueryAllStoredGameRequest) returns (QueryAllStoredGameResponse) {

option (google.api.http).get = "/alice/checkers/checkers/stored\_game";

}

}

proto /

checkers /

query.proto

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/stored-game/proto/checkers/query.proto" \l "L15-L35" \t "_blank)

Ignite CLI separates concerns into different files in the compilation of a service. Some should be edited and some should not. The following were prepared by Ignite CLI for your checkers game:

* The [query parameters (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/blob/stored-game/x/checkers/types/query.pb.go#L196-L198), as well as [how to serialize (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/blob/stored-game/x/checkers/types/query.pb.go#L741) and make them conform to the right Protobuf [RequestQuery (opens new window)↗](https://github.com/tendermint/tendermint/blob/331860c/abci/types/types.pb.go#L750-L755) interface.
* The primary implementation of the gRPC service.
* The implementation of all the storage [setters and getters (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/blob/stored-game/x/checkers/keeper/grpc_query_stored_game.go) as extra functions in the keeper.
* The implementation of the storage getters in the keeper [as they come from the gRPC server (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/blob/stored-game/x/checkers/keeper/grpc_query_stored_game.go).

[#Copy link](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html#additional-helper-functions) Additional helper functions

Your stored game's black and red fields are only strings, but they represent sdk.AccAddress or even a game from the rules file. Therefore, add helper functions to StoredGame to facilitate operations on them. Create a new file x/checkers/types/full\_game.go.

1. Get the game's black player:



Copy

func (storedGame StoredGame) GetBlackAddress() (black sdk.AccAddress, err error) {

black, errBlack := sdk.AccAddressFromBech32(storedGame.Black)

return black, sdkerrors.Wrapf(errBlack, ErrInvalidBlack.Error(), storedGame.Black)

}

x /

checkers /

types /

full\_game.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/full-game-object/x/checkers/types/full_game.go" \l "L12-L15" \t "_blank)

Note how it introduces a new error ErrInvalidBlack, which you define shortly. Do the same for the [red (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/blob/full-game-object/x/checkers/types/full_game.go#L17-L20) player.

1. Parse the game so that it can be played. The Turn has to be set by hand:



Copy

func (storedGame StoredGame) ParseGame() (game \*rules.Game, err error) {

board, errBoard := rules.Parse(storedGame.Board)

if errBoard != nil {

return nil, sdkerrors.Wrapf(errBoard, ErrGameNotParseable.Error())

}

board.Turn = rules.StringPieces[storedGame.Turn].Player

if board.Turn.Color == "" {

return nil, sdkerrors.Wrapf(fmt.Errorf("turn: %s", storedGame.Turn), ErrGameNotParseable.Error())

}

return board, nil

}

x /

checkers /

types /

full\_game.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/full-game-object/x/checkers/types/full_game.go" \l "L22-L32" \t "_blank)

1. Add a function that checks a game's validity:



Copy

func (storedGame StoredGame) Validate() (err error) {

\_, err = storedGame.GetBlackAddress()

if err != nil {

return err

}

\_, err = storedGame.GetRedAddress()

if err != nil {

return err

}

\_, err = storedGame.ParseGame()

return err

}

x /

checkers /

types /

full\_game.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/full-game-object/x/checkers/types/full_game.go" \l "L34-L45" \t "_blank)

1. Introduce your own errors:



Copy

var (

ErrInvalidBlack = sdkerrors.Register(ModuleName, 1100, "black address is invalid: %s")

ErrInvalidRed = sdkerrors.Register(ModuleName, 1101, "red address is invalid: %s")

ErrGameNotParseable = sdkerrors.Register(ModuleName, 1102, "game cannot be parsed")

)

x /

checkers /

types /

errors.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/full-game-object/x/checkers/types/errors.go" \l "L11-L14" \t "_blank)

[#Copy link](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html#unit-tests) Unit tests

Now that you have added some code on top of what Ignite CLI created for you, you should add unit tests. You will not add code to test the code generated by Ignite CLI, as your project is not yet ready to *test the framework*. However, Ignite CLI added some unit tests of its own. Run those for the keeper:

**Local**

**Docker**



Copy

$ go test github.com/alice/checkers/x/checkers/keeper

Copy

$ docker run --rm -it \

-v $(pwd):/checkers \

-w /checkers \

checkers\_i \

go test github.com/alice/checkers/x/checkers/keeper

It should pass and return something like:



Copy

ok github.com/alice/checkers/x/checkers/keeper 0.083s

[#Copy link](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html#your-first-unit-test) Your first unit test

A good start is to test that the default genesis is created as expected. Ignite already created a unit test for the genesis in [x/checkers/types/genesis\_test.go (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/blob/stored-game/x/checkers/types/genesis_test.go). It runs simple validity tests on different genesis examples.

Take your time to understand how it works, as this testing pattern is reused elsewhere. [Three cases (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/blob/stored-game/x/checkers/types/genesis_test.go#L16-L53) are tested: [case 1 (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/blob/stored-game/x/checkers/types/genesis_test.go#L16-L20), [case 2 (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/blob/stored-game/x/checkers/types/genesis_test.go#L21-L39), and [case 3 (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/blob/stored-game/x/checkers/types/genesis_test.go#L40-L53). In each case, there is a [made-up genesis object (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/blob/stored-game/x/checkers/types/genesis_test.go#L13), an [expected validity result (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/blob/stored-game/x/checkers/types/genesis_test.go#L14), and [some text (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/blob/stored-game/x/checkers/types/genesis_test.go#L12) to help the reader make sense of it. This [array of cases (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/blob/stored-game/x/checkers/types/genesis_test.go#L11) is then run through the [test proper (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/blob/stored-game/x/checkers/types/genesis_test.go#L56-L63).

The unit test you add is more modest. Your test checks that the starting id on a default genesis is 1:



Copy

func TestDefaultGenesisState\_ExpectedInitialNextId(t \*testing.T) {

require.EqualValues(t,

&types.GenesisState{

StoredGameList: []types.StoredGame{},

SystemInfo: types.SystemInfo{uint64(1)},

},

types.DefaultGenesis())

}

x /

checkers /

types /

genesis\_test.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/full-game-object/x/checkers/types/genesis_test.go" \l "L67-L74" \t "_blank)

To run it, use go test with the package name:

**Local**

**Docker**



Copy

$ go test github.com/alice/checkers/x/checkers/types

Copy

$ docker run --rm -it \

-v $(pwd):/checkers \

-w /checkers \

checkers\_i \

go test github.com/alice/checkers/x/checkers/types

This should return something like:



Copy

ok github.com/alice/checkers/x/checkers/types 0.814s

Alternatively, call it from the folder itself:

**Local**

**Docker**



Copy

$ cd x/checkers/types/ && go test

Copy

$ docker run --rm -it \

-v $(pwd):/checkers \

-w /checkers/x/checkers/types \

checkers\_i \

go test



You want your tests to pass when everything is okay, but you also want them to fail when something is wrong. Make sure your new test fails by temporarily changing uint64(1) to uint64(2). You should get the following:



Copy

--- FAIL: TestDefaultGenesisState\_ExpectedInitialNextId (0.00s)

genesis\_test.go:68:

Error Trace: genesis\_test.go:68

Error: Not equal:

expected: &types.GenesisState{Params:types.Params{}, SystemInfo:types.SystemInfo{NextId:0x2}, StoredGameList:[]types.StoredGame{}}

actual : &types.GenesisState{Params:types.Params{}, SystemInfo:types.SystemInfo{NextId:0x1}, StoredGameList:[]types.StoredGame{}}

Diff:

--- Expected

+++ Actual

@@ -4,3 +4,3 @@

SystemInfo: (types.SystemInfo) {

- NextId: (uint64) 2

+ NextId: (uint64) 1

},

Test: TestDefaultGenesisState\_ExpectedInitialNextId

FAIL

FAIL github.com/alice/checkers/x/checkers/types 0.187s

FAIL

This appears complex, but the significant aspect is this:



Copy

Diff:

--- Expected

+++ Actual

- NextId: (uint64) 2

+ NextId: (uint64) 1

For *expected* and *actual* to make sense, you have to ensure that they are correctly placed in your call. When in doubt, go to the require function definition:



Copy

func EqualValues(t TestingT, expected interface{}, actual interface{}, msgAndArgs ...interface{}) {...}

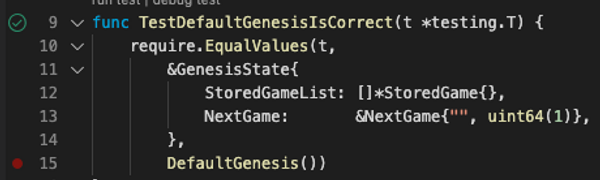
require /

require.go

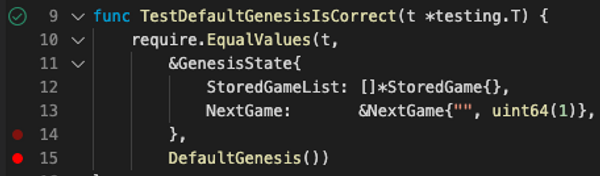
[View source→](https://github.com/stretchr/testify/blob/v1.7.0/require/require.go" \l "L202" \t "_blank)

[#Copy link](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html#debug-your-unit-test) Debug your unit test

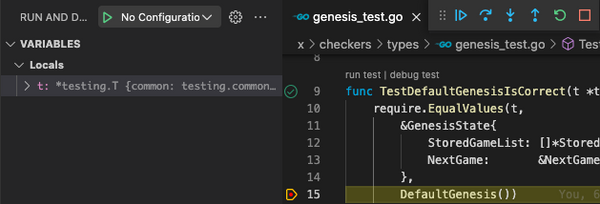
Your first unit test is a standard Go unit test. If you use an IDE like Visual Studio Code and have Go installed locally, it is ready to assist you with running the test in debug mode. Next to the function name is a small green tick or arrow. If you hover below it, a faint red dot appears:



This red dot is a potential breakpoint. Add one on the types.DefaultGenesis() line. The dot is now bright and stays there:



Right-click on the green tick, and choose Debug Test. If it asks you to install a package, accept. Eventually it stops at the breakpoint and displays the current variables and a panel for stepping actions:



If you are struggling with a test, create separate variables in order to inspect them in debug. From there, follow your regular step-by-step debugging process. If you are not familiar with debugging, [this online tutorial (opens new window)↗](https://www.digitalocean.com/community/tutorials/debugging-go-code-with-visual-studio-code) will be helpful.

[#Copy link](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html#more-unit-tests) More unit tests

With a simple yet successful unit test, you can add more consequential ones to test your helper methods.

First, create a file that declares some constants that you will reuse throughout:



Copy

package testutil

const (

Alice = "cosmos1jmjfq0tplp9tmx4v9uemw72y4d2wa5nr3xn9d3"

Bob = "cosmos1xyxs3skf3f4jfqeuv89yyaqvjc6lffavxqhc8g"

Carol = "cosmos1e0w5t53nrq7p66fye6c8p0ynyhf6y24l4yuxd7"

)

x /

checkers /

testutil /

constants.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/full-game-object/x/checkers/testutil/constants.go" \t "_blank)

Create a new file x/checkers/types/full\_game\_test.go and declare it in [package types\_test (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/blob/full-game-object/x/checkers/types/full_game_test.go#L1). Since you are going to repeat some actions, it is worth adding a reusable function:



Copy

const (

alice = testutil.Alice

bob = testutil.Bob

)

func GetStoredGame1() types.StoredGame {

return types.StoredGame{

Black: alice,

Red: bob,

Index: "1",

Board: rules.New().String(),

Turn: "b",

}

}

x /

checkers /

types /

full\_game\_test.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/full-game-object/x/checkers/types/full_game_test.go" \l "L13-L27" \t "_blank)

Now you can test the function to get the black player's address. One test for the happy path, and another for the error:



Copy

func TestCanGetAddressBlack(t \*testing.T) {

aliceAddress, err1 := sdk.AccAddressFromBech32(alice)

black, err2 := GetStoredGame1().GetBlackAddress()

require.Equal(t, aliceAddress, black)

require.Nil(t, err2)

require.Nil(t, err1)

}

func TestGetAddressWrongBlack(t \*testing.T) {

storedGame := GetStoredGame1()

storedGame.Black = "cosmos1jmjfq0tplp9tmx4v9uemw72y4d2wa5nr3xn9d4" // Bad last digit

black, err := storedGame.GetBlackAddress()

require.Nil(t, black)

require.EqualError(t,

err,

"black address is invalid: cosmos1jmjfq0tplp9tmx4v9uemw72y4d2wa5nr3xn9d4: decoding bech32 failed: invalid checksum (expected 3xn9d3 got 3xn9d4)")

require.EqualError(t, storedGame.Validate(), err.Error())

}

x /

checkers /

types /

full\_game\_test.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/full-game-object/x/checkers/types/full_game_test.go" \l "L29-L46" \t "_blank)

You can do the same for [Red (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/blob/full-game-object/x/checkers/types/full_game_test.go#L48-L65).

Test that [you can parse a game (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/blob/full-game-object/x/checkers/types/full_game_test.go#L67-L71), even [if it has been tampered with (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/blob/full-game-object/x/checkers/types/full_game_test.go#L73-L79), except [if the tamper is wrong (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/blob/full-game-object/x/checkers/types/full_game_test.go#L81-L88) or [if the turn is wrongly saved (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/blob/full-game-object/x/checkers/types/full_game_test.go#L90-L97).

Also make sure that a default test created by Ignite CLI is correct in using the default values of SystemInfo instead of erasing them:



Copy

func networkWithSystemInfoObjects(t \*testing.T) (\*network.Network, types.SystemInfo) {

...

- systemInfo := &types.SystemInfo{}

+ systemInfo := state.SystemInfo

nullify.Fill(&systemInfo)

...

- return network.New(t, cfg), \*state.SystemInfo

+ return network.New(t, cfg), state.SystemInfo

}

x /

checkers /

... /

cli /

query\_system\_info\_test.go

[View source→](https://github.com/cosmos/b9-checkers-academy-draft/blob/full-game-object/x/checkers/client/cli/query_system_info_test.go" \l "L24-L30" \t "_blank)

Interested in integration tests? Skip ahead to the [section](https://ida.interchain.io/hands-on-exercise/2-ignite-cli-adv/5-integration-tests.html) where you learn about them.

[#Copy link](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html#interact-via-the-cli) Interact via the CLI

Ignite CLI created a set of files for you. It is time to see whether you can already interact with your new checkers blockchain.

1. Start the chain in its shell:

**Local**

**Docker**



Copy

$ ignite chain serve --reset-once

Copy

$ docker run --rm -it \

--name checkers \

-v $(pwd):/checkers \

-w /checkers \

checkers\_i \

ignite chain serve --reset-once

The throwaway container is started with the name checkers, so that you can connect to it for the next commands.

This ends and holds with:



Copy

...

🌍 Tendermint node: http://0.0.0.0:26657

🌍 Blockchain API: http://0.0.0.0:1317

🌍 Token faucet: http://0.0.0.0:4500

1. Check the values saved in SystemInfo. Look at the relevant client/cli file, which Ignite CLI created to find out what command is relevant. Here it is [query\_system\_info.go (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/blob/stored-game/x/checkers/client/cli/query_system_info.go#L14). You can also ask the CLI:

**Local**

**Docker**



Copy

$ checkersd query checkers --help

Copy

$ docker exec -it checkers \

checkersd query checkers --help

Note how it connects to the newly created container named checkers.

Which returns something like:



Copy

Available Commands:

...

show-system-info shows systemInfo

Therefore, you call it:

**Local**

**Docker**



Copy

$ checkersd query checkers show-system-info

Copy

$ docker exec -it checkers \

checkersd query checkers show-system-info

This returns:



Copy

SystemInfo:

nextId: "1"

This is as expected. No games have been created yet, so the game counter is still at 1.



You may encounter an error like the following:



Copy

command not found checkersd

This indicates that you likely have not configured Go correctly. Refer back to GOPATH in our [Go introduction](https://ida.interchain.io/tutorials/4-golang-intro/1-install.html).

1. The --output flag allows you to get your results in a JSON format, which might be useful if you would like to use a script to parse the information. When you use the --help flag, you see which flags are available for a specific command:

**Local**

**Docker**



Copy

$ checkersd query checkers show-system-info --help

Copy

$ docker exec -it checkers \

checkersd query checkers show-system-info --help

Among the output, you see:



Copy

...

-o, --output string Output format (text|json) (default "text")

Now try again a bit differently:

**Local**

**Docker**



Copy

$ checkersd query checkers show-system-info --output json

Copy

$ docker exec -it checkers \

checkersd query checkers show-system-info --output json

This should print:



Copy

{"SystemInfo":{"nextId":"1"}}

1. You can similarly confirm there are no [stored games (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/blob/stored-game/x/checkers/client/cli/query_stored_game.go#L14):

**Local**

**Docker**



Copy

$ checkersd query checkers list-stored-game

Copy

$ docker exec -it checkers \

checkersd query checkers list-stored-game

This should print:



Copy

pagination:

next\_key: null

total: "0"

storedGame: []

Remember how you wrote --no-message? That was to not create messages or transactions, which would directly update your checkers storage. Soft-confirm there are no commands available:

**Local**

**Docker**



Copy

$ checkersd tx checkers --help

Copy

$ docker exec -it checkers \

checkersd tx checkers --help

synopsis

To summarize, this section has explored:

* How to begin creating an original blockchain application, in this case a checkers game, identifying and prioritizing the basic core game features to build a foundation for future improvements.
* How to define a checkers rule set by searching for and obtaining an existing implementation, rather than needlessly duplicating complex coding work.
* The minimum game information it is necessary to store, and how to store it making use of Ignite CLI.
* The Protobuf objects created by Ignite CLI, which locates objects in different files depending on their use and updates them to include informative comments indicating where code has been added.
* The files created by Ignite CLI which you can and should update, for example by setting the default genesis values.
* The Protobuf services and service interfaces created by Ignite CLI that you will fill with objects and messages when scaffolding a new module.
* How to add helper functions which you can add to perform operations on the strings that represent your stored games, such as getting the game creator and players and introducing your own errors.
* How to add, run, and debug unit tests to check the functionality of your code.
* How to use Ignite CLI to confirm that you can interact with your new checkers blockchain.

previous

[](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/2-exercise-intro.html)

**[Exercise - Make a Checkers Blockchain](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/2-exercise-intro.html)**

up next

**[Create Custom Messages](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/4-create-message.html)**

[[](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/4-create-message.html)](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/4-create-message.html)

Rate this Page

icon smile

icon meh

icon frown

Would you like to add a message?

Submit

Thank you for your Feedback!

[](https://ida.interchain.io/ida-course/discord-info.html)

On this page

[Some initial thoughts](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html#some-initial-thoughts)

[Code needs](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html#code-needs)

[Defining the rule set](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html#defining-the-rule-set)

[The stored game object](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html#the-stored-game-object)

[How to store](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html#how-to-store)

[Looking around](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html#looking-around)

[Protobuf objects](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html#protobuf-objects)

[How Ignite CLI works](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html#how-ignite-cli-works)

[Files to adjust](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html#files-to-adjust)

[Protobuf service interfaces](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html#protobuf-service-interfaces)

[Additional helper functions](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html#additional-helper-functions)

[Unit tests](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html#unit-tests)

[Your first unit test](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html#your-first-unit-test)

[Debug your unit test](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html#debug-your-unit-test)

[More unit tests](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html#more-unit-tests)

[Interact via the CLI](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/3-stored-game.html#interact-via-the-cli)

#### **Get Cosmos updates**

Unsubscribe at any time. [Privacy Policy↗](https://v1.cosmos.network/privacy)

     Next

Documentation

[Cosmos SDK](https://docs.cosmos.network/)[Cosmos Hub](https://hub.cosmos.network/)[CometBFT](https://docs.cometbft.com/)[IBC Protocol](https://ibc.cosmos.network/)

Community

[Interchain blog](https://blog.cosmos.network/)[Forum](https://forum.cosmos.network/)[Discord](https://discord.gg/cosmosnetwork)

Contributing

[Source code on GitHub](https://github.com/cosmos/sdk-tutorials)

[](https://ida.interchain.io/)

[Interchain Developer Academy](https://ida.interchain.io/)

**[](https://blog.cosmos.network/)[](https://twitter.com/cosmos)[](https://discord.gg/cosmosnetwork)[](https://www.linkedin.com/company/interchain-foundation/about/)[](https://reddit.com/r/cosmosnetwork)[](https://t.me/cosmosproject)[](https://www.youtube.com/c/CosmosProject)**



Dark mode

† This website is maintained by the Interchain Foundation (ICF). The contents and opinions of this website are those of the ICF. The ICF provides links to cryptocurrency exchanges as a service to the public. The ICF does not warrant that the information provided by these websites is correct, complete, and up-to-date. The ICF is not responsible for their content and expressly rejects any liability for damages of any kind resulting from the use, reference to, or reliance on any information contained within these websites.

Cosmos is a registered trademark of the [Interchain Foundation.](https://interchain.io/)[Privacy](https://v1.cosmos.network/privacy)