**Section recap**

**Building a Provably Fair Lottery on the Blockchain**

Today, let's do a recap of a project we recently accomplished — creating a provably fair lottery using blockchain technology! This might sound complex (and it is!), but it's exciting news. Let's understand why.

Ever thought of why you should use any other lottery system that's not blockchain-based? To be honest, the answer is a definite 'No.' The most compelling reason being that no other lottery provides you with the same level of randomness and transparency as blockchain does.

So, buckle up! Let's dive into this tutorial and take apart every component of our blockchain lottery contract.

**Custom Errors, Enums and Private Variables**

In our lottery contract, we kicked things off with some custom gas-efficient errors, including errors that accept multiple parameters. Part of the code we utilized was the type declarations. We took advantage of enums, assigning them different values and wrapping them as unsigned integers.

One essential part of our lottery contract was our beautiful, private state variables—part of our strong privacy practice. Additionally, we created getters for any variables we wanted to expose.

**Verbose Constructor**

One particular feature is that the constructor of our contract is verbose. By adjusting the deployment parameters accordingly, we are able to deploy this contract on any chain, ensuring flawless functionality. This holds true whether we're forking a testnet or a mainnet. The only thing required to accommodate a different network is a distinct network configuration.

**Raffle and Chainlink Automation**

We designed a raffle that emits a log, streamlining migrations and frontend indexing. We also learned about and integrated the Chainlink automation to automatically call our smart contracts.

The automation upkeep of our smart contracts led to an amazing result—it ran once because the perform upkeep requirement was met.

**Smart Contract Execution and Testing**

Once triggered, the Chainlink network replies by calling the fulfillRandomWords function, which selects our random winner. We got a good look into the CEI - checks effects interactions pattern, where we implement checks, conduct effects and eventually process our external interactions outside of the smart contracts.

We provided several getter functions. Surprisingly, the codebase for this project is only about 200 lines long, but it felt much longer because of the advanced scripting and deployment methods we had to learn.

We deployed our contract using a helper configuration file. This ensured that this deployment will function flawlessly on whatever chain it's deployed. To bridge interactions with actual blockchain, while testing, we deployed mock contracts.

**Broadcasting and Interaction via Command Line**

Once the mockup and initial stage were completed, we began broadcasting and deploying our Raffle. Subsequently, we added our consumer to the VRF programmatically.

Additionally, we devised an interactions script to make it easier for us to run commands for adding consumers, creating, or funding subscriptions from our command line. This is way more straightforward than having to interact with cast.

**Testing and Debugging**

During the process, we wrote comprehensive unit tests, though we intentionally left some areas that you can explore to level up your skill sets. For example, we tested with mock Chainlink tokens and learned some exciting testing techniques like capturing the event outputs to reuse later in our tests.

We also worked a lot with modifiers and expected a revert with this abi encoder thing. Understanding that will be a task for later.

Finally, we deployed this lottery on an actual testnet chain, funding our automation subscription and our VRF subscription with Link. We observed Chainlink nodes handling all this with no issues.

**Recap**

This has been a massive project, filled with learnings and hands-on coding experience. If you've followed through, congratulate yourself. This is the perfect time to take a break, soak up some sun, or binge on your favorite treats.

Remember to post about this amazing project on Twitter, link it to your GitHub and give yourself a pat on the back. Progressing this far is a significant achievement.

As we advance through this course, you can broaden your technical knowledge related to the Web3 ecosystem. I look forward to your being a part of the remaining exciting lessons in this course. Till then, happy coding!

**Congratulations**

You've completed the course! You're now ready to build your own blockchain applications. Now you are ready to delve into advanced chapters of this course. Take a break, and then come back to learn more about the Web3 ecosystem.