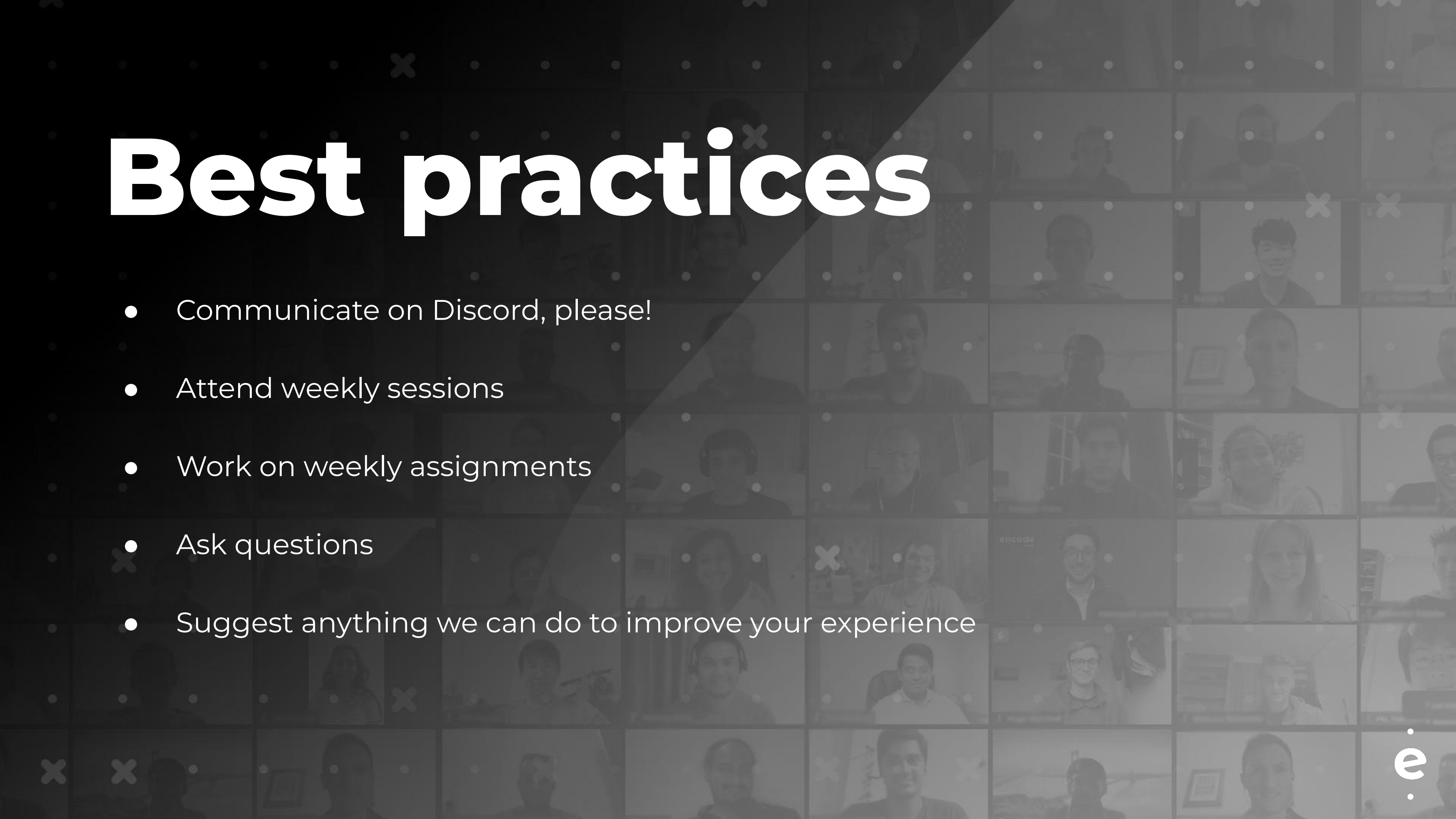
encode CLUB

Algorand Bootcamp

Before we start

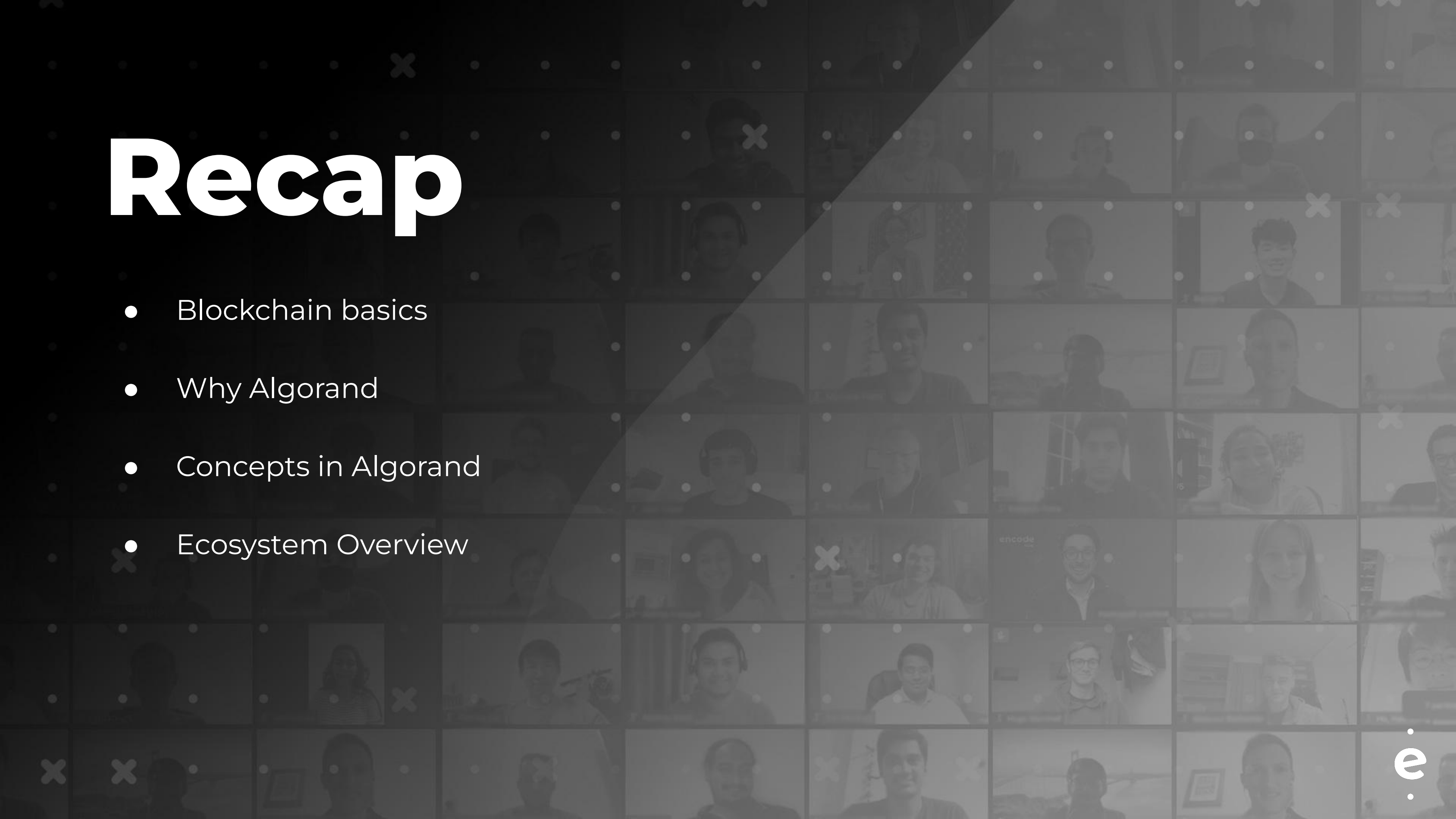
Any questions from last week?

How to make the best use of bootcamp?



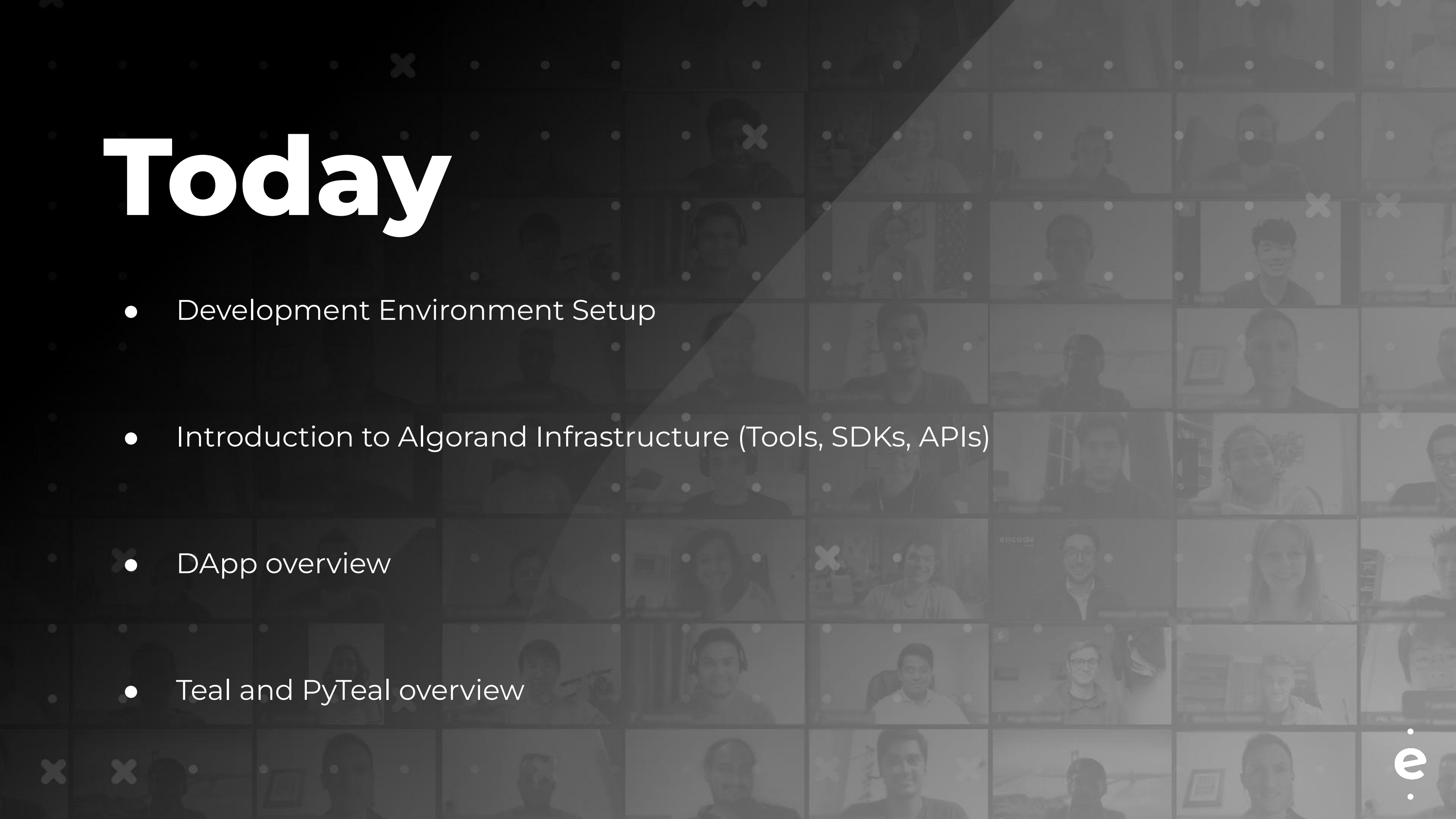
Best practices

- Communicate on Discord, please!
- Attend weekly sessions
- Work on weekly assignments
- Ask questions
- Suggest anything we can do to improve your experience



Recap

- Blockchain basics
- Why Algorand
- Concepts in Algorand
- Ecosystem Overview



Today

- Development Environment Setup
- Introduction to Algorand Infrastructure (Tools, SDKs, APIs)
- DApp overview
- Teal and PyTeal overview

Dev Environment Setup

Algorand Networks

- Mainnet:
  - ALGO and Real Assets are traded

- TestNet:
  - Test Applications with realistic network conditions prior to deploying them on MainNet

- Betanet:
  - Access the newest protocol-level features

Connect to Node

- Sandbox: (**Easy**)
  - The sandbox allows developers to create local, private networks. Moreover, you can quickly remove a network, reset its state, or spin up a new network

- Third-party API Services: (**Very, very, very easy**)
  - This is an excellent choice if you don't want to set up a local network using Docker, and just want to experiment with Algorand development initially

Connect to Node

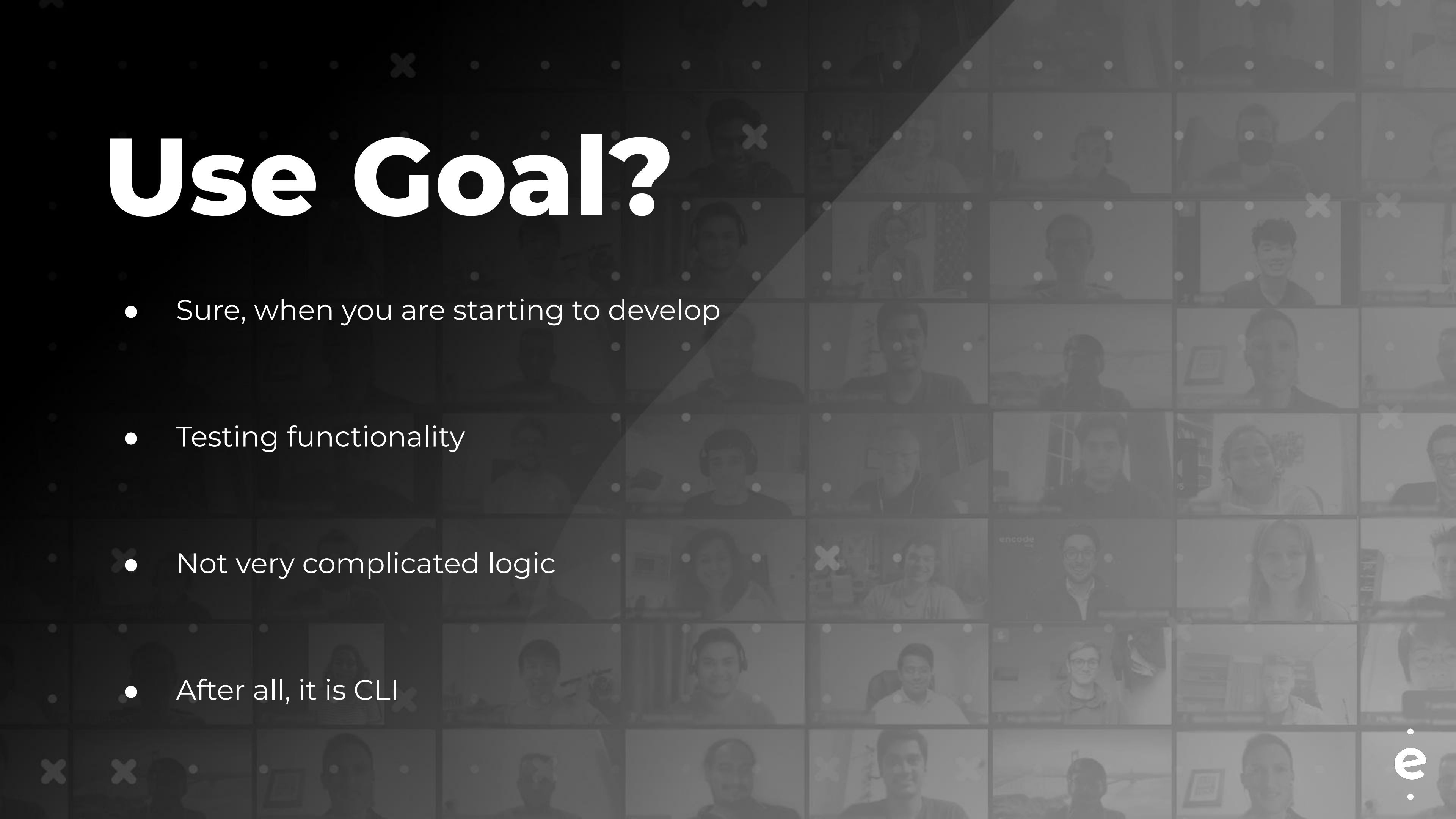
- Run your own node: (**difficult**)
  - You can decide to **<u>run your own Algorand node</u>**, which contains the full implementation of the Algorand software.

Interaction

- GOAL CLI
- Python, JavaScript, Go, Java SDKs
- REST APIs

Goal CLI

- GOAL is the CLI for interacting Algorand software instance.
- The binary 'goal' is installed alongside the algod binary and is considered an integral part of the complete installation.
- The binaries should be used in tandem - you should not try to use a version of goal with a different version of algod.

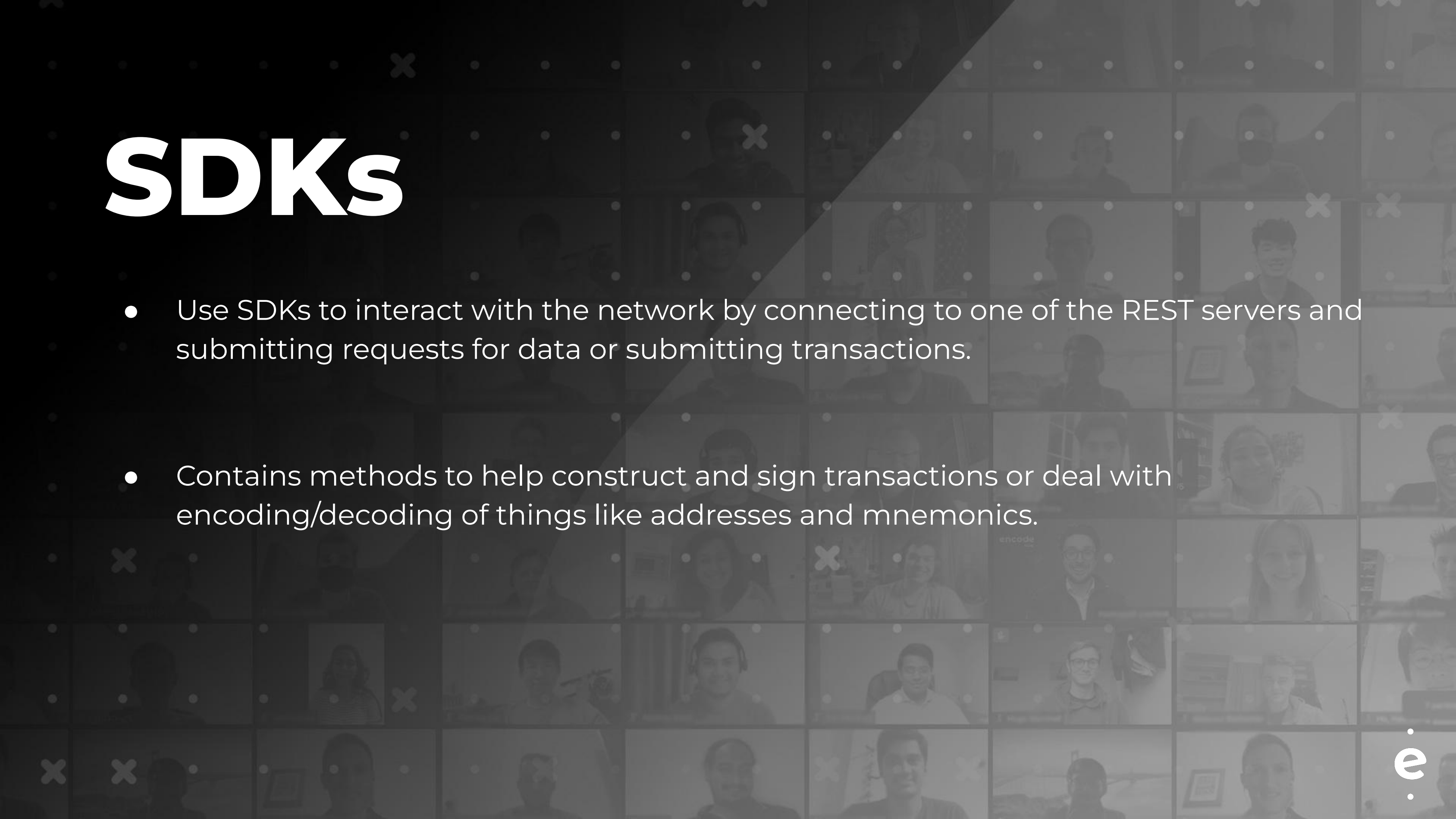


Use Goal?

- Sure, when you are starting to develop
- Testing functionality
- Not very complicated logic
- After all, it is CLI

More resources

- goal - Algorand Developer Portal
- goal account - Algorand Developer Portal
- goal app - Algorand Developer Portal
- goal asset - Algorand Developer Portal
- goal network - Algorand Developer Portal



SDKs

- Use SDKs to interact with the network by connecting to one of the REST servers and submitting requests for data or submitting transactions.

- Contains methods to help construct and sign transactions or deal with encoding/decoding of things like addresses and mnemonics.

Demo

Concepts Overview

- Algorand Standard Assets
- Smart Contracts
- Smart Signatures

ASAs

- On-chain assets
- Benefit from the same security, compatibility, speed and ease of use as the Algo
- Can represent stablecoins, loyalty points, system credits, and in-game points

Assets Overview

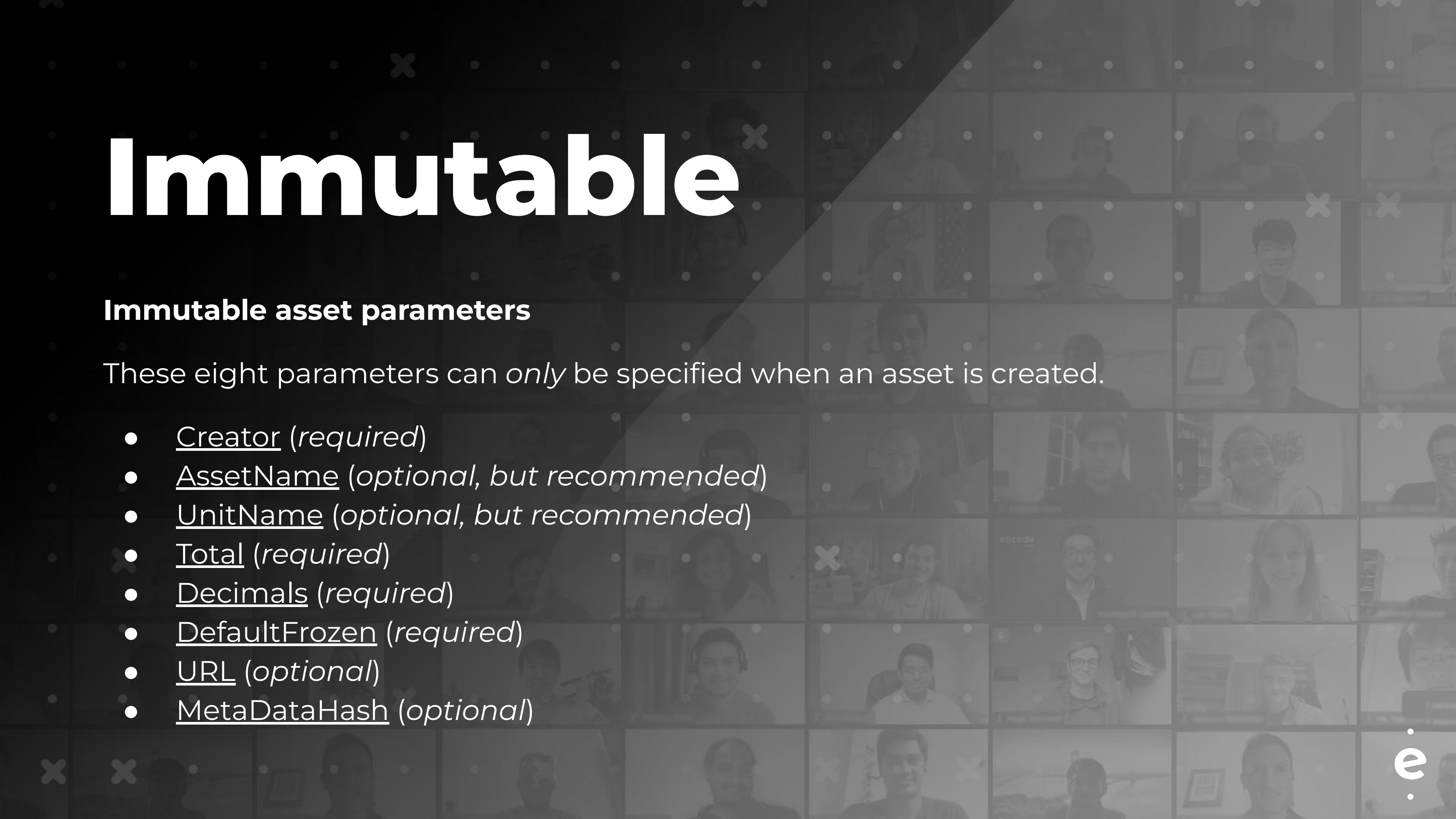
- For every asset an account creates or owns, its minimum balance is increased by 0.1 Algos (100,000 microAlgos).

- Before a new asset can be transferred to a specific account the receiver must opt-in to receive the asset. This process is described below in Receiving an asset.

- If any transaction is issued that would violate the minimum balance requirements, the transaction will fail.

Asset Parameters

- Immutable Parameters - 8
- Mutable Asset Parameters - 4



Immutable

Immutable asset parameters

These eight parameters can *only* be specified when an asset is created.

- Creator (*required*)
- AssetName (*optional, but recommended*)
- UnitName (*optional, but recommended*)
- Total (*required*)
- Decimals (*required*)
- DefaultFrozen (*required*)
- URL (*optional*)
- MetaDataHash (*optional*)

Mutable

- Manager Address
  - The manager account is the only account that can authorize transactions to re-configure or destroy an asset.
- Reserve Address
  - Specifying a reserve account signifies that non-minted assets will reside in that account instead of the default creator account.

Mutable

- Freeze Address
  - The freeze account is allowed to freeze or unfreeze the asset holdings for a specific account.

- Clawback Address
  - The clawback address represents an account that is allowed to transfer assets from and to any asset holder (assuming they have opted-in).

Assets Demo

Smart Contracts

- Algorand Smart Contracts (ASC1) are small programs that serve various functions on the blockchain and operate on layer-1.

- Smart contracts are separated into two main categories, smart contracts, and smart signatures.

- These types are also referred to as stateful and stateless contracts respectively.

Smart Contracts

- Applications can modify state associated with the application

- Applications can access on-chain values (account balances, asset config parameters, etc.)

Smart Contracts

- Applications can execute transactions as part of execution of the logic (InnerTxns)

- Applications can have an associated Application Account that can hold ALGOs or ASAs

Smart Contracts

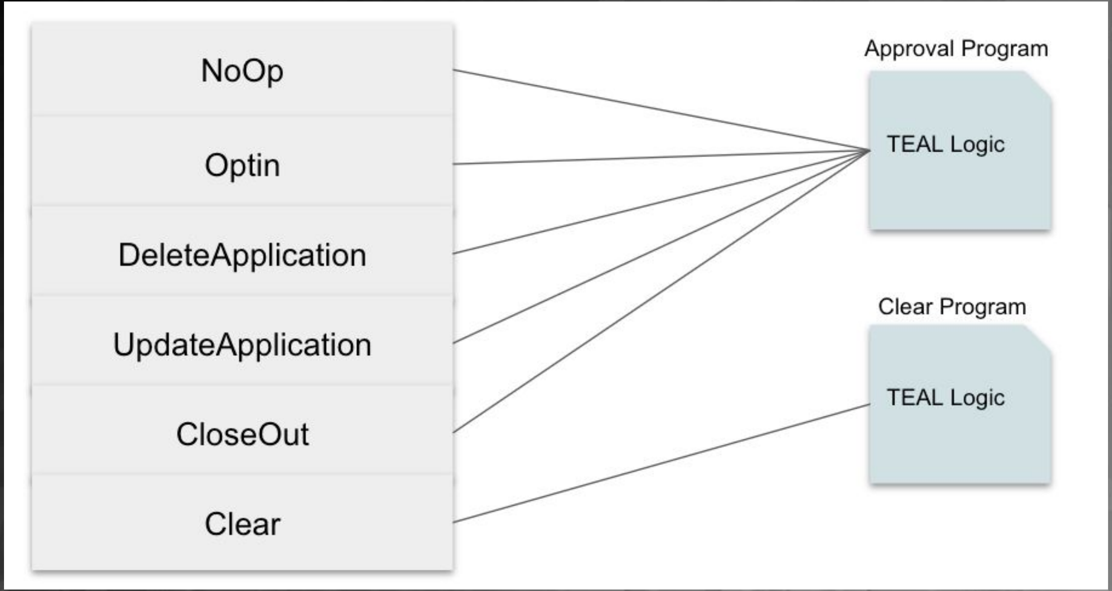
- Approval Program
  - Processing application calls to the contract
  - Implementing most of the logic of an application

- Clear State Program
  - Handle accounts using the clear call to remove the smart contract from their balance record

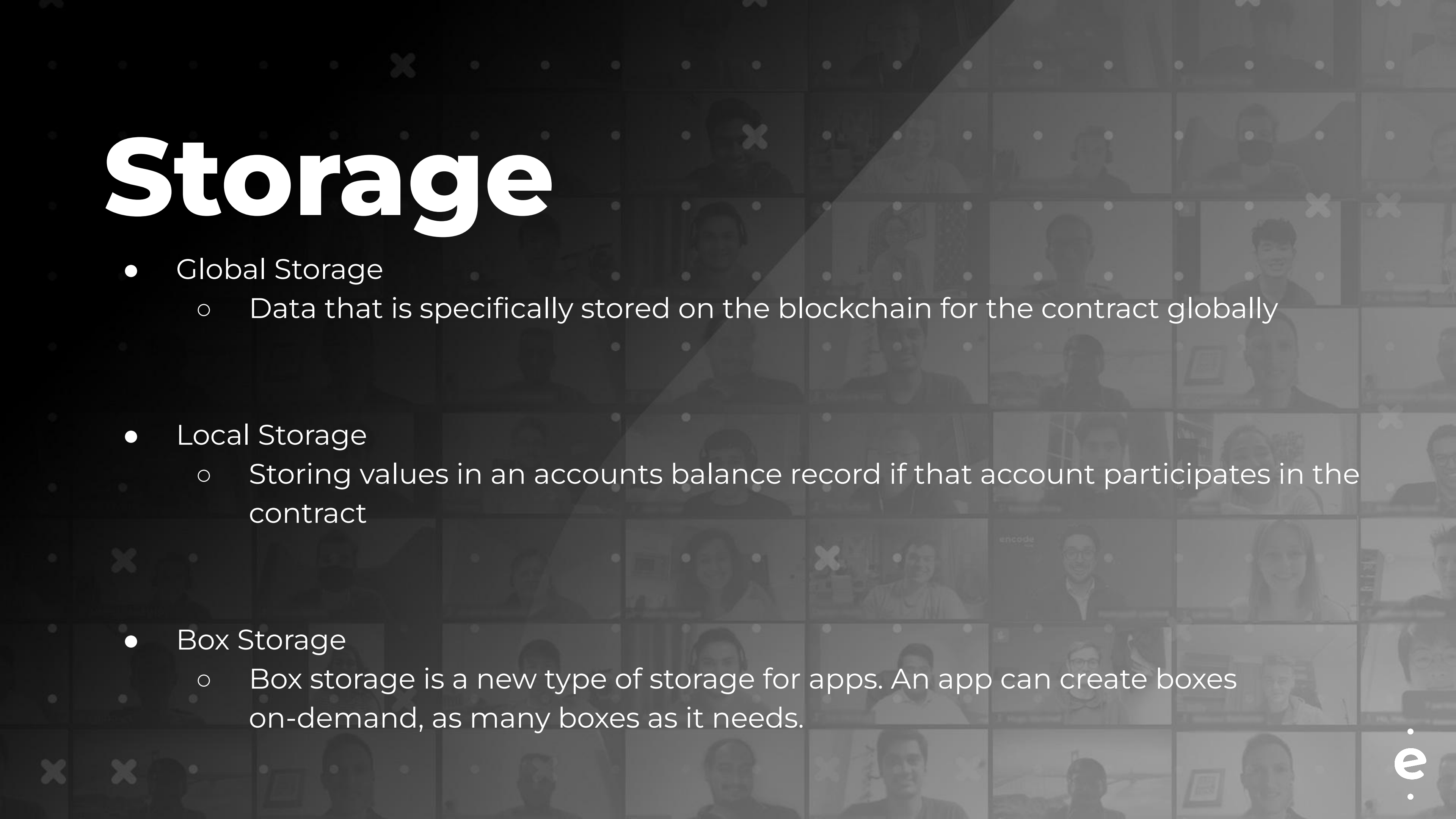
Application Call Txns

- NoOp
- OptIn
- Delete Application
- Update Application
- CloseOut
- ClearState

Smart Contract



Source: https://developer.algorand.org/docs/get-details/dapps/smart-contracts/apps/



Storage

- Global Storage
  - Data that is specifically stored on the blockchain for the contract globally
- Local Storage
  - Storing values in an accounts balance record if that account participates in the contract
- Box Storage
  - Box storage is a new type of storage for apps. An app can create boxes on-demand, as many boxes as it needs.

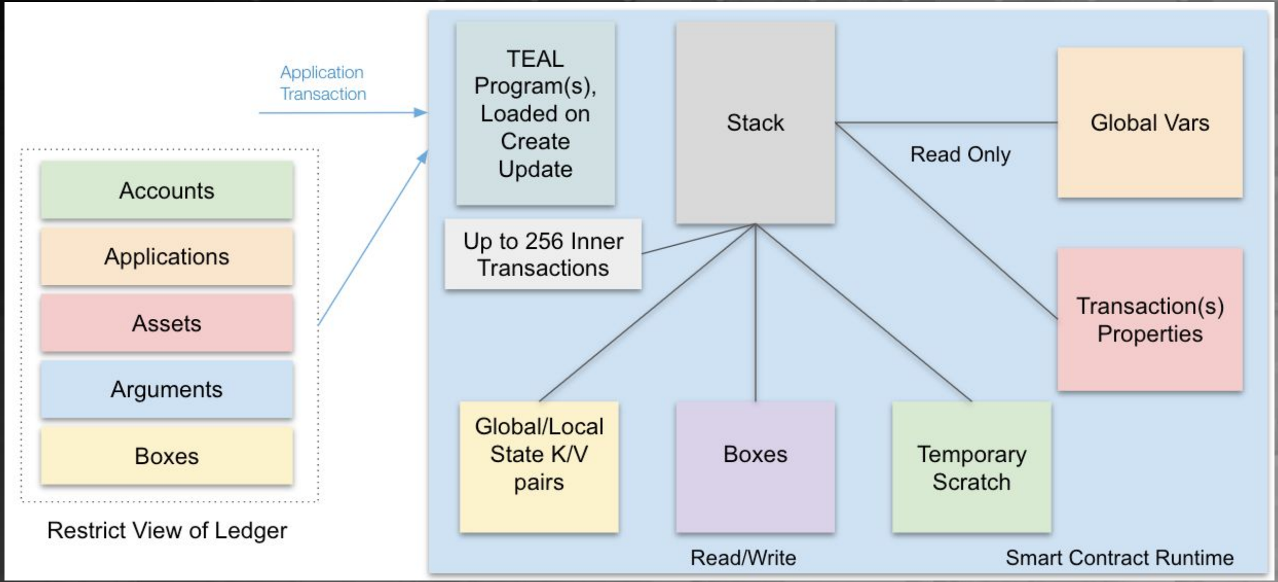
Arrays in SC

- Applications Array
- Accounts Array
- Assets Array

Arrays in SC

- Arguments Arrays
- Box Array

Arrays



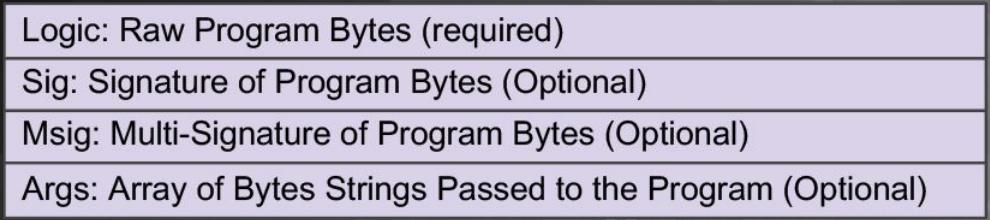
Source: https://developer.algorand.org/docs/get-details/dapps/smart-contracts/apps/

Smart Sig

- Primarily used to delegate signature authority.
- Smart signatures can also be used as escrow or contract accounts

Smart Sig

- Can be used as either a contract account, or for delegated approval



Logic: Raw Program Bytes (required)
Sig: Signature of Program Bytes (Optional)
Msig: Multi-Signature of Program Bytes (Optional)
Args: Array of Bytes Strings Passed to the Program (Optional)

Source: Smart Sig Modes

How to code?

- TEAL (Transaction Execution Approval Language)
- PyTeal
- Reach Lang

Next Week

- Pyteal overview
- Data Types and Constants
- Arithmetic and Byte Operators
- Transaction Fields
- Global Parameters
- Scratch Space

Assignment

1. Review the slides and recordings of this week
2. Create an Algorand Standard Asset **ON TESTNET** with an SDK of your choice
3. Perform all the asset related transactions on the asset created in step 2
   a. Reference to asset operations: [Link]
4. ***Organize the code in such a way that you only need one python script to perform all operations (including the initial funding transaction)***

Resources

1. Testnet Funds Dispenser: Link
2. AlgoNode APIs: Link

Thank You

Email: hi@lalithmedury.com

Twitter: @LalithMedury

Web: https://lalithmedury.com

Email: info@encode.club

Twitter: @encodeclub

Web: https://encode.club