CESS

Cumulus Encrypted Storage System

CESS Course Week 3 - Episode 6

Develop Your Own Pallets

CESS Official Website

Table of Content

CESS
Cumulus Encrypted Storage System

What is building your own pallets about?

CESS
Cumulus Encrypted Storage System

- Fork the cess core (github)
- Customizing the logic in the source code, e.g adding pallets, modifying runtime parameters, adding other codes
- Run the cess-node as a separate chain

Why build your own pallets?

CESS
Cumulus Encrypted Storage System

Pros:

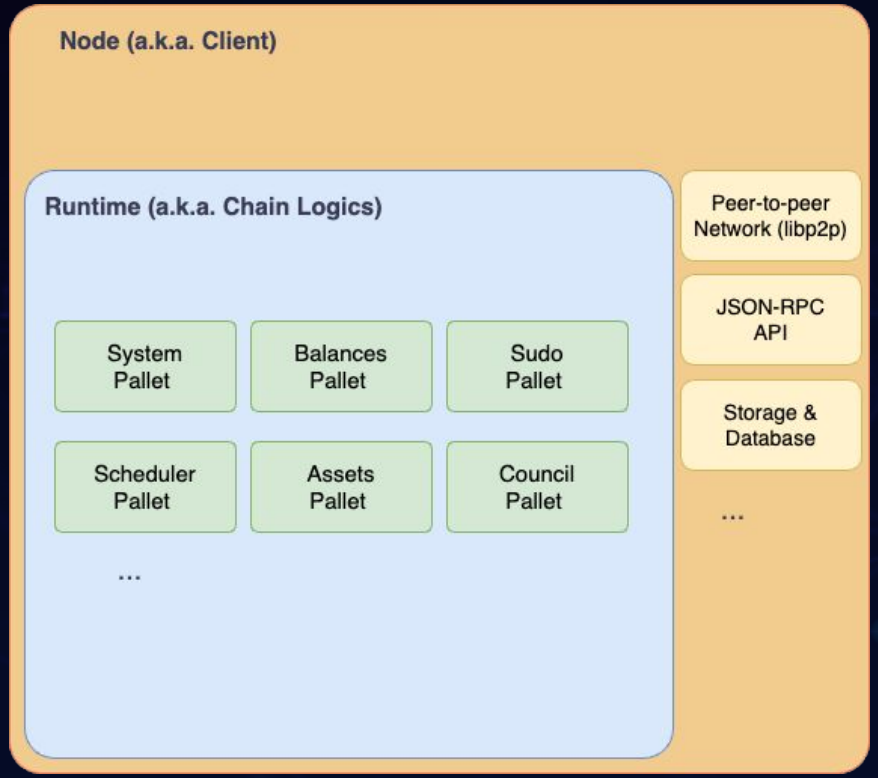
- Flexible and the most powerful
- Can adjust from consensus mechanism, transaction fee, to block time.

Cons:

- A high technical barrier
  - learning Rust and Substrate macro
  - Familiarize with existing codebase
- Running your own chain requires more devOps and evening maintaining validator community.

High Level Description of CESS (Substrate) Framework

CESS
Cumulus Encrypted Storage System

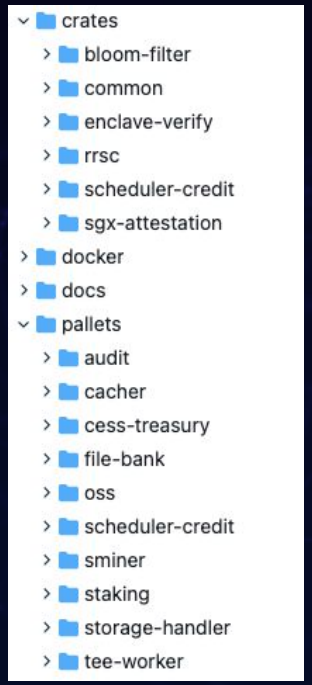


- **Client**: p2p networking, low-level library inclusion of (merkle tree) storage & database, JSON-RPC API
- **Runtime:** a composition of pallets connected together to form the core chain logics and expose the extrinsics that are publicly available.
- **Pallets**: modules that perform a specific set of functions on-chain.

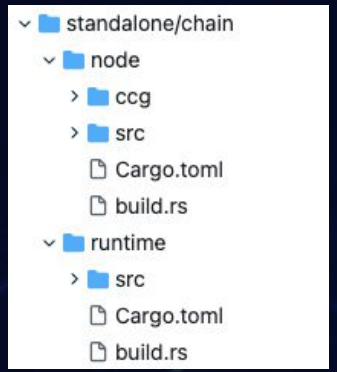
Example: CESS Node

CESS
Cumulus Encrypted Storage System

src: https://github.com/CESSProject/cess/blob/main



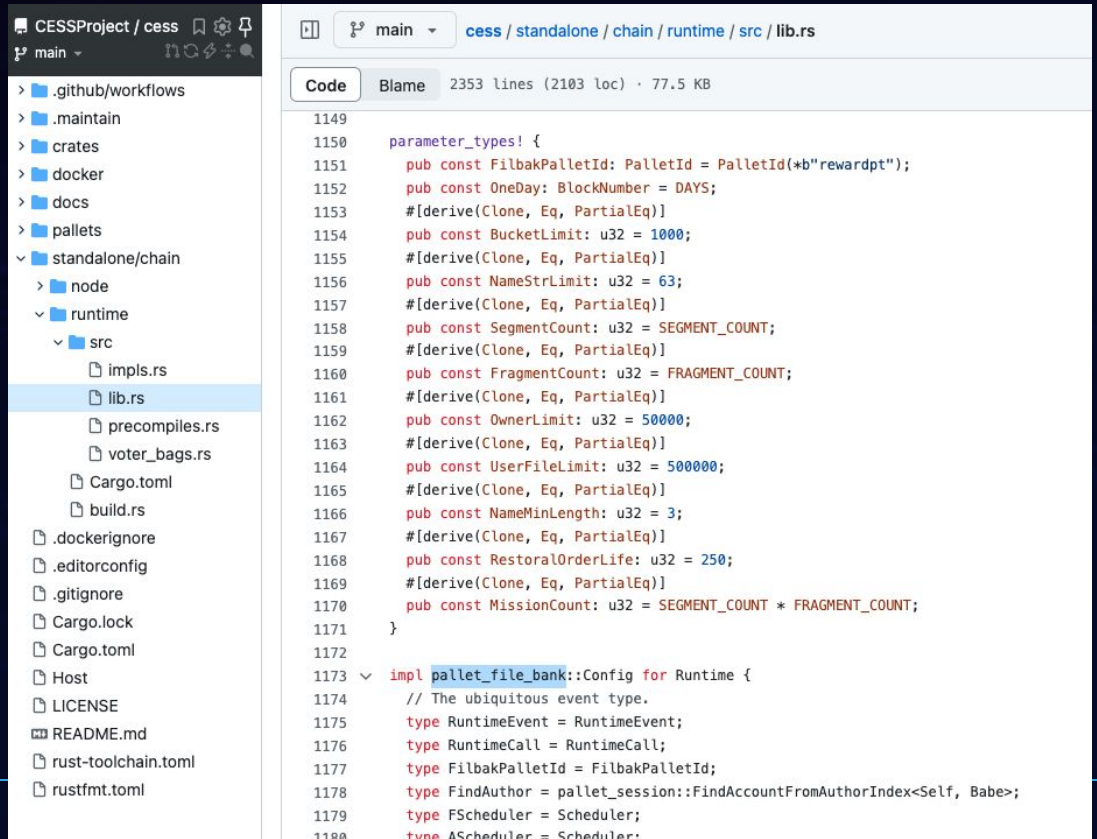
Pallets and **libraries** (crates)



node (client) and **runtime**

How Are Pallets Assembled Together?

CESS
Cumulus Encrypted Storage System



Runtime implements the **pallet::Config**

How Are Pallets Assembled Together?

CESS
Cumulus Encrypted Storage System

```
1442    parameter_types! {
1443      pub BlockGasLimit: U256 = U256::from(BLOCK_GAS_LIMIT);
1444      pub const GasLimitPovSizeRatio: u64 = BLOCK_GAS_LIMIT.saturating_div(MAX_POV_SIZE);
1445      pub PrecompilesValue: FrontierPrecompiles<Runtime> = FrontierPrecompiles::<_>::new();
1446      pub WeightPerGas: Weight = Weight::from_parts(weight_per_gas(BLOCK_GAS_LIMIT, NORMAL_DIS
1447    }
1448
1449    impl pallet_evm::Config for Runtime {
1450      type FeeCalculator = BaseFee;
1451      type GasWeightMapping = pallet_evm::FixedGasWeightMapping<Self>;
1452      type WeightPerGas = WeightPerGas;
1453      type BlockHashMapping = pallet_ethereum::EthereumBlockHashMapping<Self>;
1454      type CallOrigin = EnsureAddressTruncated;
1455      type WithdrawOrigin = EnsureAddressTruncated;
1456      type AddressMapping = HashedAddressMapping<BlakeTwo256>;
1457      type Currency = Balances;
1458      type RuntimeEvent = RuntimeEvent;
1459      type PrecompilesType = FrontierPrecompiles<Self>;
1460      type PrecompilesValue = PrecompilesValue;
1461      type ChainId = EVMChainId;
1462      type BlockGasLimit = BlockGasLimit;
1463      type Runner = pallet_evm::runner::stack::Runner<Self>;
1464      type OnChargeTransaction = ();
1465      type OnCreate = ();
1466      type FindAuthor = FindAuthorTruncated<Babe>;
1467      type GasLimitPovSizeRatio = GasLimitPovSizeRatio;
1468      type Timestamp = Timestamp;
1469      type WeightInfo = pallet_evm::weights::SubstrateWeight<Self>;
1470    }
```

Even the EVM capability in CESS chain node is setup this way.

How Are Pallets Assembled Together?

CESS
Cumulus Encrypted Storage System

```
1616  // Create the runtime by composing the FRAME pallets that were previously configured.
1617  construct_runtime!(
1618      pub enum Runtime
1619      {
1620          // Basic stuff
1621          System: frame_system = 0,
1622          RandomnessCollectiveFlip: pallet_insecure_randomness_collective_flip = 1,
1623          Timestamp: pallet_timestamp = 2,
1624          Sudo: pallet_sudo = 3,
1625          Scheduler: pallet_scheduler = 4,
1626          Preimage: pallet_preimage = 5,
1627          Mmr: pallet_mmr = 6,
1628
1629          // Account lookup
1630          // ...
1631
1632          // Tokens & Fees
1633          Balances: pallet_balances = 10,
1634          TransactionPayment: pallet_transaction_payment = 11,
1635          Assets: pallet_assets = 12,
1636          AssetTxPayment: pallet_asset_tx_payment = 13,
1637
1638          // Consensus
1639          // ...
1640
1641          // Governance
1642          // ...
1643
1644          // Smart contracts
1645          Contracts: pallet_contracts = 50,
1646          Ethereum: pallet_ethereum = 51,
1647          EVM: pallet_evm = 52,
1648          EVMChainId: pallet_evm_chain_id = 53,
1649          DynamicFee: pallet_dynamic_fee = 54,
1650          BaseFee: pallet_base_fee = 55,
1651
1652          // CESS pallets
1653          // ...
1654          FileBank: pallet_file_bank = 60,
1655      }
1656  );
```

Put the pallet inside the **`construct_runtime!{ ... }`** macro

Pallet Structure: `pallet-file-bank`

CESS
Cumulus Encrypted Storage System

src: https://github.com/cessProject/cess/blob/main/pallets/file-bank/src/lib.rs

```rust
#[frame_support::pallet]
pub mod pallet {
    // ...

    #[pallet::config]
    pub trait Config: frame_system::Config + sp_std::fmt::Debug {
        /// The overarching event type.
        type RuntimeEvent: From<Event<Self>> + IsType<<Self as frame_system::Config>::RuntimeEvent>;

        type WeightInfo: WeightInfo;

        type RuntimeCall: From<Call<Self>>;

        type FScheduler: ScheduleNamed<BlockNumberFor<Self>, Self::SProposal, Self::SPalletsOrigin>;

        type AScheduler: ScheduleAnon<BlockNumberFor<Self>, Self::SProposal, Self::SPalletsOrigin>;
        /// Overarching type of all pallets origins.
        type SPalletsOrigin: From<frame_system::RawOrigin<Self::AccountId>>;
        // ...

        /// pallet address.
        #[pallet::constant]
        type FilbakPalletId: Get<PalletId>;

        #[pallet::constant]
        type UserFileLimit: Get<u32> + Clone + Eq + PartialEq;

        #[pallet::constant]
        type OneDay: Get<BlockNumberFor<Self>>;

        // User defined name length limit
        #[pallet::constant]
        type NameStrLimit: Get<u32> + Clone + Eq + PartialEq;
        // Maximum number of containers that users can create.
        #[pallet::constant]
        type BucketLimit: Get<u32> + Clone + Eq + PartialEq;

        // ...
    }
```

- **Pallet Config**
- Pallet Events
- Pallet Errors
- Pallet Storage
- Pallet Lifecycle Callbacks
- Pallet Extrinsics
- Pallet Helper Functions

Pallet Structure: `pallet-file-bank`

CESS
Cumulus Encrypted Storage System

src: https://github.com/cessProject/cess/blob/main/pallets/file-bank/src/lib.rs

```rust
#[pallet::event]
#[pallet::generate_deposit(pub(super) fn deposit_event)]
pub enum Event<T: Config> {
    //file upload declaration
    UploadDeclaration { operator: AccountOf<T>, owner: AccountOf<T>, deal_hash: Hash },
    //file uploaded.
    TransferReport { acc: AccountOf<T>, deal_hash: Hash },
    //File deletion event
    DeleteFile { operator:AccountOf<T>, owner: AccountOf<T>, file_hash: Hash },
    // ...
}
```

- Pallet Config
- **Pallet Events**
- Pallet Errors
- Pallet Storage
- Pallet Lifecycle Callbacks
- Pallet Extrinsics
- Pallet Helper Functions

Pallet Structure: `pallet-file-bank`

CESS
Cumulus Encrypted Storage System

src: https://github.com/cessProject/cess/blob/main/pallets/file-bank/src/lib.rs

```rust
#[pallet::error]
pub enum Error<T> {
    Existed,

    FileExistent,
    //file doesn't exist.
    FileNonExistent,
    //overflow.
    Overflow,

    NotOwner,

    NotQualified,
    //It is not an error message for scheduling operation
    ScheduleNonExistent,
    // ...
}
```

- Pallet Config
- Pallet Events
- **Pallet Errors**
- Pallet Storage
- Pallet Lifecycle Callbacks
- Pallet Extrinsics
- Pallet Helper Functions

Pallet Structure: `pallet-file-bank`

CESS
Cumulus Encrypted Storage System

src: https://github.com/cessProject/cess/blob/main/pallets/file-bank/src/lib.rs

```rust
#[pallet::storage]
#[pallet::getter(fn deal_map)]
pub(super) type DealMap<T: Config> = StorageMap<_, Blake2_128Concat, Hash, DealInfo<T>>;

#[pallet::storage]
#[pallet::getter(fn file)]
pub(super) type File<T: Config> =
    StorageMap<_, Blake2_128Concat, Hash, FileInfo<T>>;

#[pallet::storage]
#[pallet::getter(fn user_hold_file_list)]
pub(super) type UserHoldFileList<T: Config> = StorageMap<
    _,
    Blake2_128Concat,
    T::AccountId,
    BoundedVec<UserFileSliceInfo, T::UserFileLimit>,
    ValueQuery,
>;

#[pallet::storage]
#[pallet::getter(fn miner_lock)]
pub(super) type MinerLock<T: Config> =
    StorageMap<_, Blake2_128Concat, AccountOf<T>, BlockNumberFor<T>>;

// ...
```

- Pallet Config
- Pallet Events
- Pallet Errors
- **Pallet Storage**
- Pallet Lifecycle Callbacks
- Pallet Extrinsics
- Pallet Helper Functions

Pallet Structure: `pallet-file-bank`

CESS
Cumulus Encrypted Storage System

src: https://github.com/cessProject/cess/blob/main/pallets/file-bank/src/lib.rs

```rust
#[pallet::hooks]
impl<T: Config> Hooks<BlockNumberFor<T>> for Pallet<T> {
    fn on_initialize(now: BlockNumberFor<T>) -> Weight {
        let days = T::OneDay::get();
        let mut weight: Weight = Weight::zero();
        // FOR TESTING
        if now % days == 0u32.saturated_into() {
            let (temp_weight, acc_list) = T::StorageHandle::frozen_task();
            weight = weight.saturating_add(temp_weight);
            let temp_acc_list: BoundedVec<AccountOf<T>, ConstU32<5000>> =
                acc_list.try_into().unwrap_or_default();
            ClearUserList::<T>::put(temp_acc_list);
            weight = weight.saturating_add(T::DbWeight::get().writes(1));
        }

        let mut count: u32 = 0;
        let acc_list = ClearUserList::<T>::get();
        weight = weight.saturating_add(T::DbWeight::get().reads(1));
        for acc in acc_list.iter() {
            // todo! Delete in blocks, and delete a part of each block
            if let Ok(mut file_info_list) = <UserHoldFileList<T>>::try_get(&acc) {
                weight = weight.saturating_add(T::DbWeight::get().reads(1));
                while let Some(file_info) = file_info_list.pop() {
                    count = count.checked_add(1).unwrap_or(ONCE_MAX_CLEAR_FILE);
                    if count == ONCE_MAX_CLEAR_FILE {
                        <UserHoldFileList<T>>::insert(&acc, file_info_list);
                        return weight;
                    }
                    if let Ok(file) = <File<T>>::try_get(&file_info.file_hash) {
```

- Pallet Config
- Pallet Events
- Pallet Errors
- Pallet Storage
- **Pallet Lifecycle Callbacks**
- Pallet Extrinsics
- Pallet Helper Functions

Pallet Structure: `pallet-file-bank`

CESS
Cumulus Encrypted Storage System

src: https://github.com/cessProject/cess/blob/main/pallets/file-bank/src/lib.rs

```rust
#[pallet::call]
impl<T: Config> Pallet<T> {
    /// Upload Declaration of Data Storage
    ///
    /// This function allows a user to upload a declaration for data storage, specifying the file's metadata,
    /// deal information, and ownership details. It is used to initiate the storage process of a file.
    ///
    /// Parameters:
    /// - `origin`: The origin of the transaction.
    /// - `file_hash`: The unique hash identifier of the file.
    /// - `deal_info`: A list of segment details for data storage.
    /// - `user_brief`: A brief description of the user and the file's ownership.
    /// - `file_size`: The size of the file in bytes.
    #[pallet::call_index(0)]
    #[transactional]
    #[pallet::weight(<T as pallet::Config>::WeightInfo::upload_declaration(deal_info.len() as u32))]
    pub fn upload_declaration(
        origin: OriginFor<T>,
        file_hash: Hash,
        deal_info: BoundedVec<SegmentList<T>, T::SegmentCount>,
        user_brief: UserBrief<T>,
        file_size: u128,
    ) -> DispatchResult {
        let sender = ensure_signed(origin)?;
        // Check if you have operation permissions.
        ensure!(Self::check_permission(sender.clone(), user_brief.user.clone()), Error::<T>::NoPermission);
        // Check file specifications.
        ensure!(Self::check_file_spec(&deal_info), Error::<T>::SpecError);
        // Check whether the user-defined name meets the rules.
```

- Pallet Config
- Pallet Events
- Pallet Errors
- Pallet Storage
- Pallet Lifecycle Callbacks
- **Pallet Extrinsics**
- Pallet Helper Functions

Pallet Structure: `pallet-audit`

CESS
Cumulus Encrypted Storage System

src: https://github.com/cessProject/cess/blob/main/pallets/audit/src/lib.rs

```rust
impl<T: Config> Pallet<T> {
    /// Clear challenge data and perform associated actions for the given block number.
    ///
    /// This function is used to clear challenge data and perform various operations for a
    /// specific block number. It iterates over challenges in the `ChallengeSlip` storage item
    /// and checks the associated `ChallengeSnapShot`.
    ///
    /// # Parameters
    ///
    /// - `now`: The block number for which challenge data is to be cleared and processed.
    ///
    /// # Returns
    ///
    /// The total weight consumed by the operation.
    fn clear_challenge(now: BlockNumberFor<T>) -> Weight {
        let mut weight: Weight = Weight::zero();

        for (miner, _) in <ChallengeSlip<T>>::iter_prefix(&now) {
            if let Ok(challenge_info) = <ChallengeSnapShot<T>>::try_get(&miner) {
                weight = weight.saturating_add(T::DbWeight::get().reads(1));
                if challenge_info.prove_info.service_prove.is_none() {
                    let count = <CountedClear<T>>::get(&miner).checked_add(1).unwrap_or(6);
                    weight = weight.saturating_add(T::DbWeight::get().reads(1));

                    let _ = T::MinerControl::clear_punish(
                        &miner,
                        challenge_info.miner_snapshot.idle_space,
                        challenge_info.miner_snapshot.service_space,
                    );
                    weight = weight.saturating_add(T::DbWeight::get().reads_writes(1, 1));

                    if count >= 3 {
                        let result = T::MinerControl::force_miner_exit(&miner);
                        weight = weight.saturating_add(T::DbWeight::get().reads_writes(5, 5));
                        if result.is_err() {
                            log::info!("force clear miner: {:?} failed", miner);
                        }
```

- Pallet Config
- Pallet Events
- Pallet Errors
- Pallet Storage
- Pallet Lifecycle Callbacks
- Pallet Extrinsics
- **Pallet Helper Functions**

Pallets Integrated in CESS Node

CESS
Cumulus Encrypted Storage System

Foundational

System frame, Timestamp pallet, Sudo pallet, Scheduler pallet, etc...

Governance

Council, TechnicalCommittee, Treasury, Bounties, Multisig, etc...

Account & Fee

Proxy pallet, Indices pallet, Balances pallet, Assets pallet, etc...

Smart Contract

Contracts (ink!), Ethereum, EVM, EVMChainId, DynamicFee, etc...

Consensus

Authorship pallet, Babe pallet, Grandpa pallet, Staking pallet, etc...

CESS Specific

FileBank, TeeWorker, Audit, Sminer, StorageHandler, etc...

CESS Pallet Deep Dive: `file-bank`

CESS
Cumulus Encrypted Storage System

src: https://github.dev/cessProject/cess/blob/main/pallets/file-bank/src/lib.rs

This pallet consists of logics on managing storage space. It allows callers (users) to purchase storage space, expand the purchased space, renew the storage leases. This pallet also implements functions to CRUD (**c**reate, **r**ead, **u**pdate, **d**elete) user buckets, a concept similar to user directory. The actual files are segmented and stored in the underlying storage network, but its metadata are stored on-chain.

CESS Pallet Deep Dive: `sminer`

CESS
Cumulus Encrypted Storage System

src: https://github.dev/cessProject/cess/blob/main/pallets/sminer/src/lib.rs

`sminer` stands for Storage Miner. This pallet contains operations related to storage miners, allowing them to claim how much space it provides for how long, staking its tokens for its claimed services, and withdrawing the service provision altogether.

Demo: Adding Your Own Pallet in CESS Node

CESS
Cumulus Encrypted Storage System

1. Fork **`cess`** Github repo
2. Add a new (template) pallet in the **`pallets`** directory
3. Add the pallet in the runtime
4. Recompile cess node and execute

Example: cess-examples/cess-node

CESS
Cumulus Encrypted Storage System

End