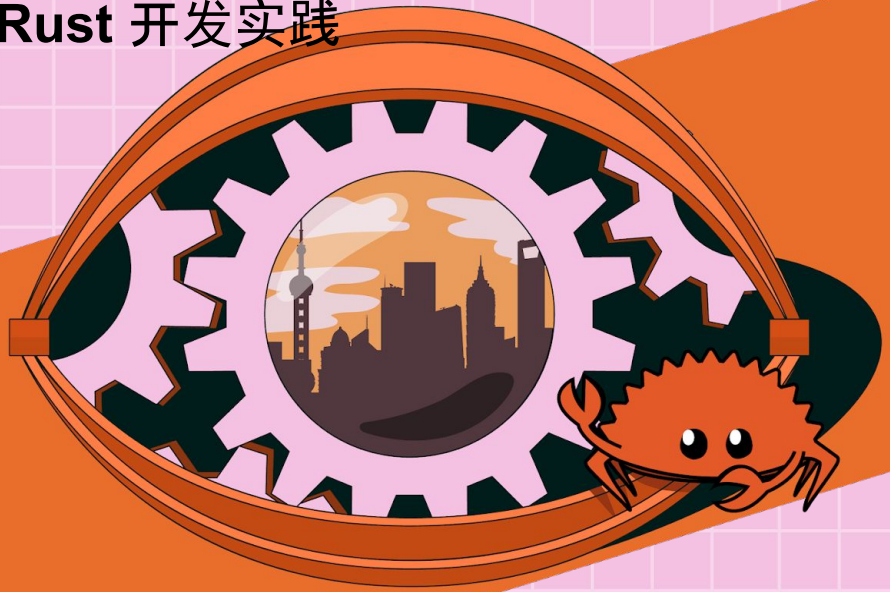
RUST CHINA CONF 2023

CRYPTAPE 祕境先锋

Axon 应用链框架的 Rust 开发实践



6.17-6.18 @Shanghai

CRYPTAPE
秘猿先锋

Who am I ?

文愿

区块链工程师@秘猿先锋

wenyuan@cryptape.com

目录

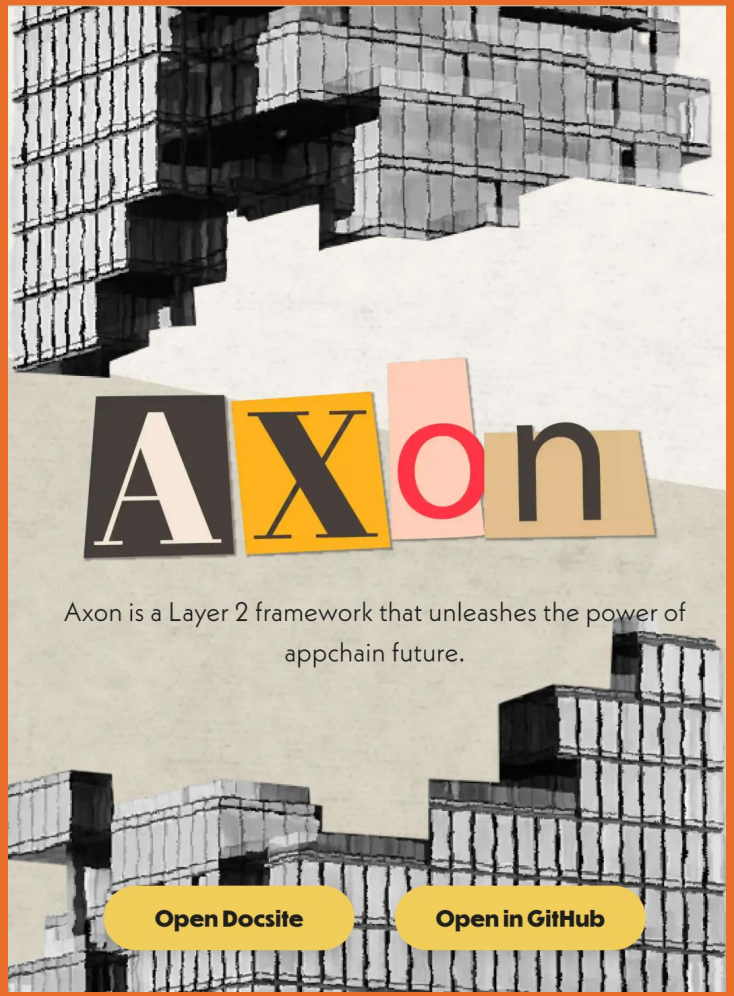
CRYPTAPE
秘境先锋

目录

CRYPTAPE
秘境先锋

What is Axon

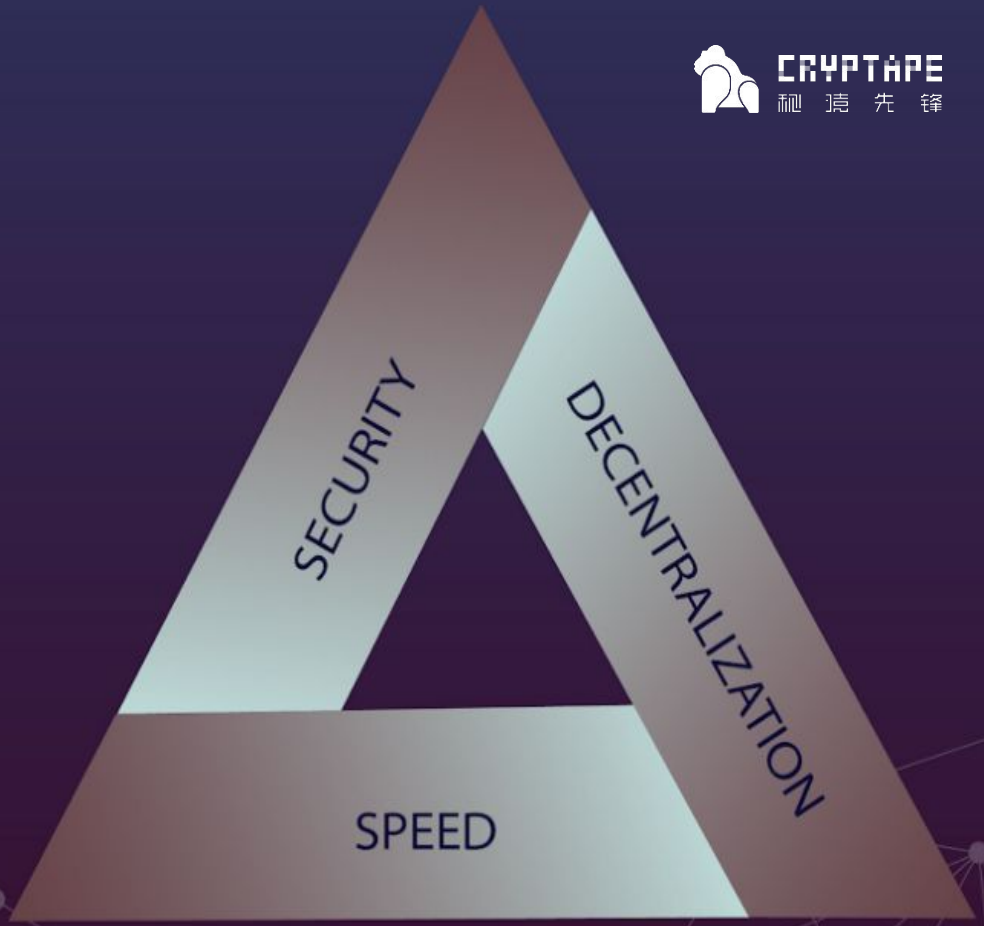
1. 应用链框架
2. 高性能
3. 互操作（Interoperability）
4. EVM 兼容
5. Rust



CRYPTAPE
秘境先锋

CRYPTAPE
秘境先锋

Blockchain Trilemma

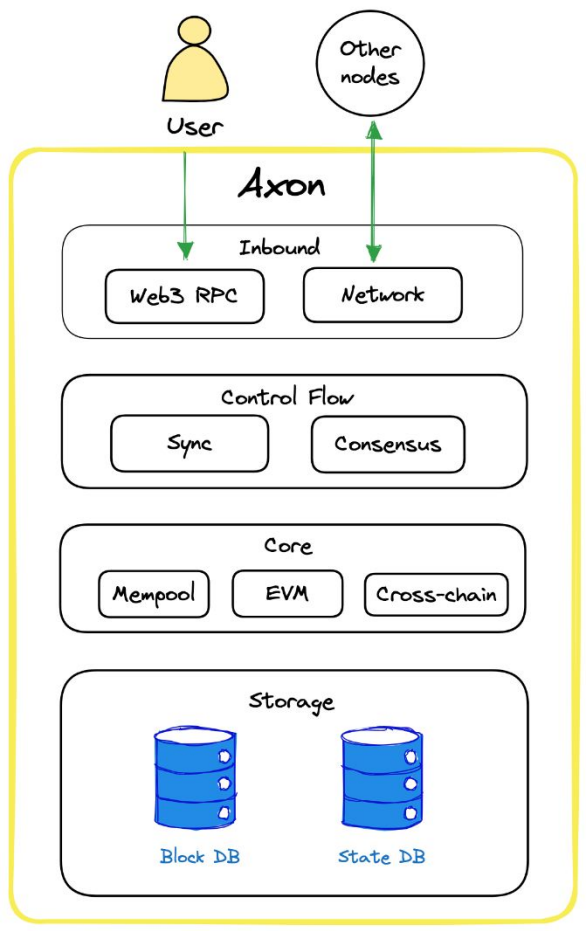


目录

CRYPTAPE
秘境先锋

主要模块

1. Mempool（交易池）
2. Consensus (Overlord)
3. P2P (Tentacle)
4. Interoperation
5. Web3 RPC（以太坊兼容）
6. Storage (KV 数据库)
7. Executor



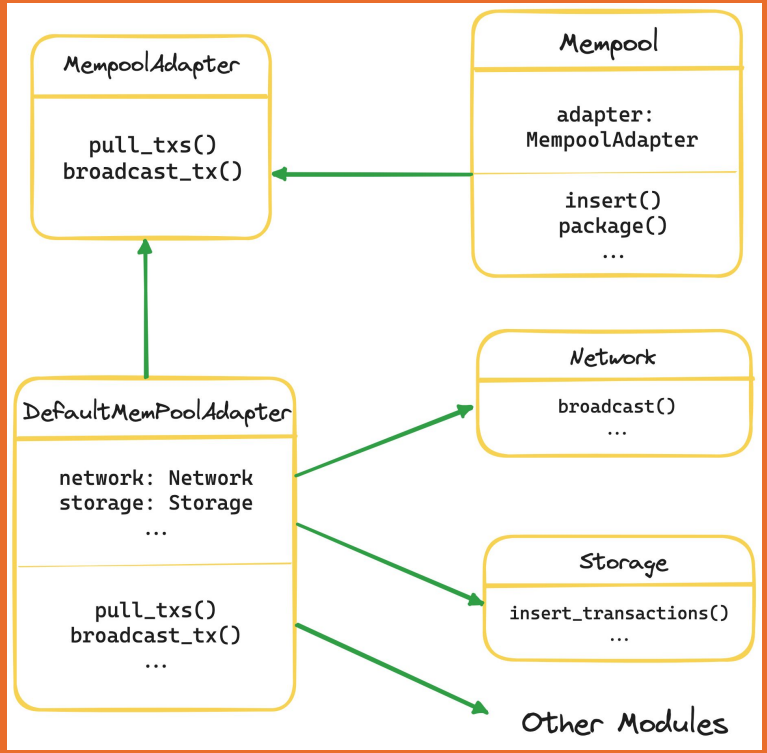
大型项目的开发难点

CRYPTAPE
祕境先锋

- 高复杂性，组件和子系统较多，相互依赖和交互，整体结构和逻辑非常复杂，开发、测试、调试难度大
- 可维护性，开发完成后，各个模块需要维护、升级和改进
- 可扩展性
- 高性能
- 高并发
- 高可靠性
- 代码管理
- …

优点

- 抽象
- 便于协作开发
- 易于测试



Mempool Traits

CRYPTAPE 祕境先锋

```rust
#[async_trait]
pub trait MemPool: Send + Sync {
    async fn insert() -> ProtocolResult<()>;
    async fn package() -> ProtocolResult<PackedTxHashes>;
    // ...
}

You, 1 second ago | 2 authors (You and others)
#[async_trait]
pub trait MemPoolAdapter: Send + Sync {
    async fn pull_txs() -> ProtocolResult<Vec<SignedTransaction>>;
    async fn broadcast_tx() -> ProtocolResult<()>;
    async fn check_authorization() -> ProtocolResult<U256>;
    // ...
}
```

实现

CRYPTAPE 秘境先锋

```rust
pub struct DefaultMemPoolAdapter<C, N, S, DB, I> {
    network:  N,
    storage:  Arc<S>,
    trie_db:  Arc<DB>,
    metadata: Arc<MetadataHandle>,

    addr_nonce:  DashMap<H160, (U256, U256)>,
    gas_limit:   AtomicU64,
    max_tx_size: AtomicUsize,
    chain_id:    u64,

    stx_tx: UnboundedSender<(Option<usize>, SignedTransaction)>,
    err_rx: Mutex<UnboundedReceiver<ProtocolError>>,

    pin_c: PhantomData<C>,
    pin_i: PhantomData<I>,
}
```

实现

```rust
async fn check_authorization(
    &self,
    ctx: Context,
    tx: &SignedTransaction,
) -> ProtocolResult<U256> {
    if is_call_system_script(tx.transaction.unsigned.action()) {…

    let addr: &H160 = &tx.sender;
    if let Some(res: Ref<'_, H160, (U256, U256)>) = self.addr_nonce.get(addr) {
        if tx.transaction.unsigned.nonce() < &res.value().0 {
            return Err(MemPoolError::InvalidNonce {…
            .into());
        } else if res.value().1 < tx.transaction.unsigned.may_cost() {
            return Err(MemPoolError::ExceedBalance {…
            .into());
        } else {…
    }

    let backend: AxonExecutorAdapter<S, DB> = AxonExecutorAdapter::from_root(…
    )?;

    let account: Basic = backend.basic(*addr);
    self.addr_nonce DashMap<H160, (U256, U256)>
        .insert(*addr, (account.nonce, account.balance));

    if &account.nonce > tx.transaction.unsigned.nonce() {…

    if account.balance < tx.transaction.unsigned.may_cost() {…

    Ok(tx.transaction.unsigned.nonce() - account.nonce)
} fn check_authorization
```

测试代码

CRYPTAPE 秘境先锋

```rust
pub struct HashMemPoolAdapter {
    network_txs: DashMap<Hash, SignedTransaction>,
}

impl HashMemPoolAdapter {
    fn new() -> HashMemPoolAdapter {
        HashMemPoolAdapter {
            network_txs: DashMap::new(),
        }
    }
}

#[async_trait]
impl MemPoolAdapter for HashMemPoolAdapter {
    async fn pull_txs() -> ProtocolResult<Vec<SignedTransaction>> {}
    async fn broadcast_tx() -> ProtocolResult<()> {}
    async fn check_authorization() -> ProtocolResult<U256> {
        Ok(U256::zero())
    }
    // ...
}
```

目录

CRYPTAPE
祕境先锋

如何优雅地在代码中加入埋点

CRYPTAPE
祕境先锋

- 直接调用 Prometheus API
- AOP（面向切片）
- LLVM IR
- …

■ Prometheus API

```rust
async fn send_raw_transaction(&self, tx: Hex) -> RpcResult<H256> {
    self.tx_send_total.with_label_values(&["call"]).inc();
    Ok(hash)
}
```

■ LLVM IR 插入监控埋点

1. 在 Rust 代码编译成 LLVM IR 之前解析 AST，找到需要埋点的函数
2. 为这些函数生成对应的监控指标定义代码
3. 在进入和退出这些函数的 IR 指令前插入对指标的操作 (inc 等)
4. 重新打包成 Rust 代码并编译，得到增加埋点的可执行文件

过程宏(元编程)

CRYPTAPE
祕 境 先 锋

- 优点:

1. 高度灵活
2. 零运行期成本
3. 封装性好，使用方便

- 缺点:

1. 学习曲线高
2. 可移植性差
3. 调试难度大(Cargo expand)

■ 过程宏

```rust
#[async_trait]
impl Rpc for RpcExample {
    #[metrics_rpc("eth_sendRawTransaction")]
    async fn send_transaction(&self, tx: SignedTransaction) -> Result<Hash, Error> {
        Ok(tx.transaction.hash)
    }

    #[metrics_rpc("net_listening")]
    fn listening(&self) -> Result<bool, Error> {
        Ok(false)
    }
}
```

async_wait 的展开

CRYPTAPE 祕境先锋

```
fn send_transaction<'life0, 'async_trait>(
    &'life0 self,
    tx: SignedTransaction,
) -> ::core::pin::Pin<
    Box<
        dyn ::core::future::Future<Output = Result<Hash, Error>>
            + ::core::marker::Send
            + 'async_trait,
    >,
>
where
    'life0: 'async_trait,
    Self: 'async_trait;

fn listening(&self) -> Result<bool, Error>;
```

metrics_rpc &
fut-ret

```rust
let func_block_wrapper: TokenStream = if func_return.is_ret_pin_box_fut() {
    quote! {
        Box::pin(async move {
            let inst = common_apm::Instant::now();
            let ret: #ret_ty = #func_block.await;

            if ret.is_err() {
                common_apm::metrics::api::API_REQUEST_RESULT_COUNTER_VEC_STATIC
                    .#func_ident
                    .failure
                    .inc();
                return ret;
            }

            common_apm::metrics::api::API_REQUEST_RESULT_COUNTER_VEC_STATIC
                .#func_ident
                .success
                .inc();
            common_apm::metrics::api::API_REQUEST_TIME_HISTOGRAM_STATIC
                .#func_ident
                .observe(common_apm::metrics::duration_to_sec(inst.elapsed()));

            ret
        })
    }
} else {
    quote! {
        let inst = common_apm::Instant::now();
        let ret: #func_ret_ty = #func_block;
        // ...
        ret
    }
};
```

async
send_transaction

```
Box::pin(async move {
    let inst = std::time::Instant::now();
    let ret: Result<Hash, Error> = {
        Box::pin(async move {
            if let ::core::option::Option::Some(__ret) =
                ::core::option::Option::None::<Result<Hash, Error>>
            {
                return __ret;
            }
            let __self = self;
            let tx = tx;
            let __ret: Result<Hash, Error> = { Ok(tx.transaction.hash) };
            #[allow(unreachable_code)]
            __ret
        })
    }
    .await;

    if ret.is_err() {
        common_apm::metrics::api::API_REQUEST_RESULT_COUNTER_VEC_STATIC
            .eth_sendRawTransaction
            .failure
            .inc();
        return ret;
    }

    common_apm::metrics::api::API_REQUEST_RESULT_COUNTER_VEC_STATIC
        .eth_sendRawTransaction
        .success
        .inc();
    common_apm::metrics::api::API_REQUEST_TIME_HISTOGRAM_STATIC
        .eth_sendRawTransaction
        .observe(common_apm::metrics::duration_to_sec(common_apm::elapsed(inst)));
    ret
})
```

CRYPTAPE
祕境先锋

listening

```rust
fn listening(&self) -> Result<bool, Error> {
    let inst = std::time::Instant::now();
    let ret: Result<bool, Error> = { Ok(false) };

    if ret.is_err() {
        common_apm::metrics::api::API_REQUEST_RESULT_COUNTER_VEC_STATIC
            .net_listening
            .failure
            .inc();
        return ret;
    }

    common_apm::metrics::api::API_REQUEST_RESULT_COUNTER_VEC_STATIC
        .net_listening
        .success
        .inc();
    common_apm::metrics::api::API_REQUEST_TIME_HISTOGRAM_STATIC
        .net_listening
        .observe(common_apm::metrics::duration_to_sec(common_apm::elapsed(inst)));
    ret
}
```

trace_span

CRYPTAPE 秘境先锋

```rust
#[async_trait]
impl ToTrace for TraceExample {
    #[trace_span(kind = "trace", logs = "{tx_len: txs.len()}")]
    async fn store(&self, ctx: Context, txs: Vec<SignedTransaction>) -> ProtocolResult<()> {
        debug_assert!(txs.len() == 1);
        Ok(())
    }

    #[trace_span(kind = "trace")]
    fn version(&self, ctx: Context) -> String {
        "0.1.0".to_string()
    }
}
```

version

```rust
fn version(&self, ctx: Context) -> String {
    use common_apm::tracing::{LogField, SpanContext, Tag, TRACER};
    let mut span_tags: Vec<Tag> = Vec::new();
    let mut span_logs: Vec<LogField> = Vec::new();
    let mut span = if let Some(parent_ctx) = ctx.get::<Option<SpanContext>>("parent_span_ctx") {
        if parent_ctx.is_some() {
            TRACER
                .load()
                .child_of_span("trace.version", parent_ctx.clone().unwrap(), span_tags)
        } else {
            TRACER.load().span("trace.version", span_tags)
        }
    } else {
        TRACER.load().span("trace.version", span_tags)
    };
    let ctx = match span.as_mut() {
        Some(span) => {
            span.log(|log| {
                for span_log in span_logs.into_iter() {
                    log.field(span_log);
                }
            });
            ctx.with_value("parent_span_ctx", span.context().cloned())
        }
        None => ctx,
    };
    {
        "0.1.0".to_string()
    }
}
```

目录

CRYPTAPE
秘境先锋

互操作性（Interoperability）

CRYPTAPE
秘境先锋

- CKB (Common Knowledge Base)

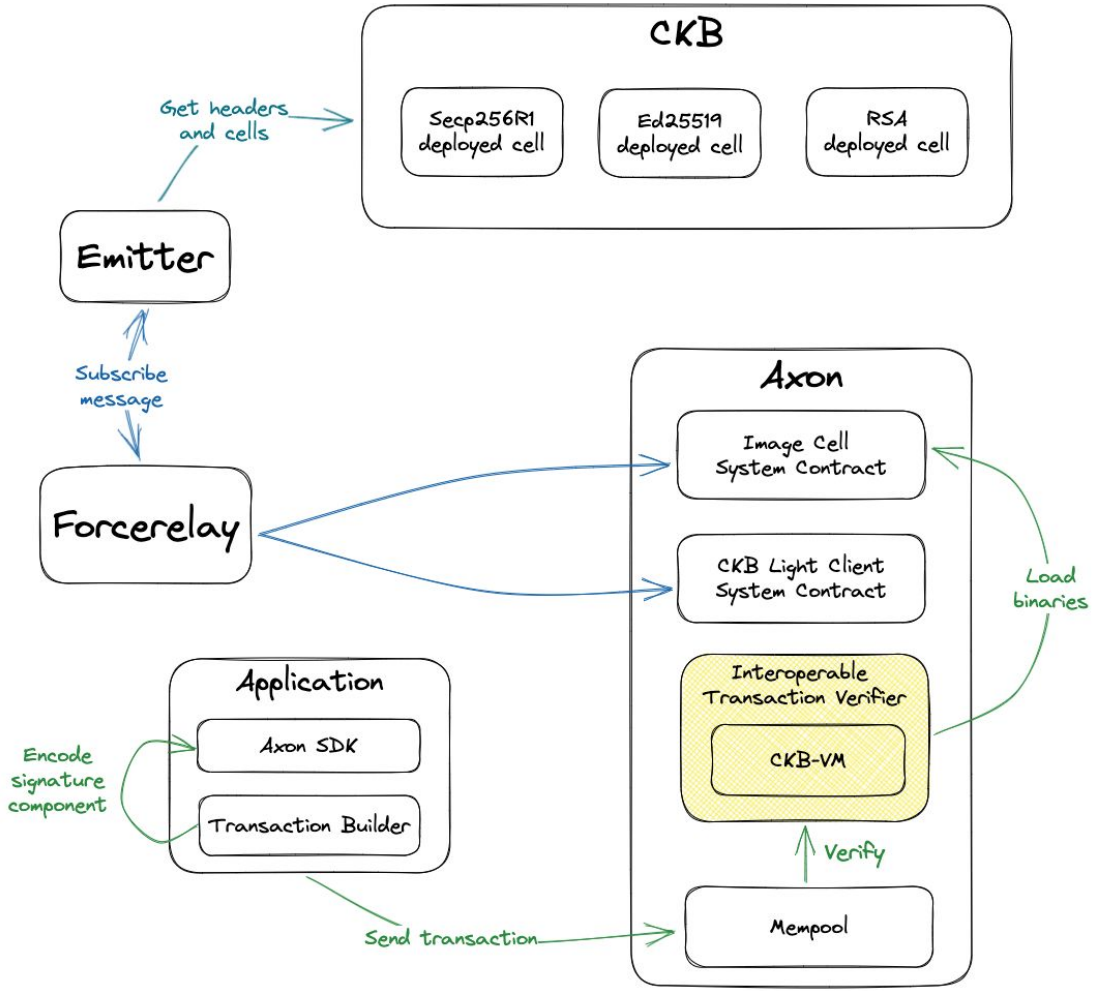
  Axon 的 Layer 1，确保安全和去中心化

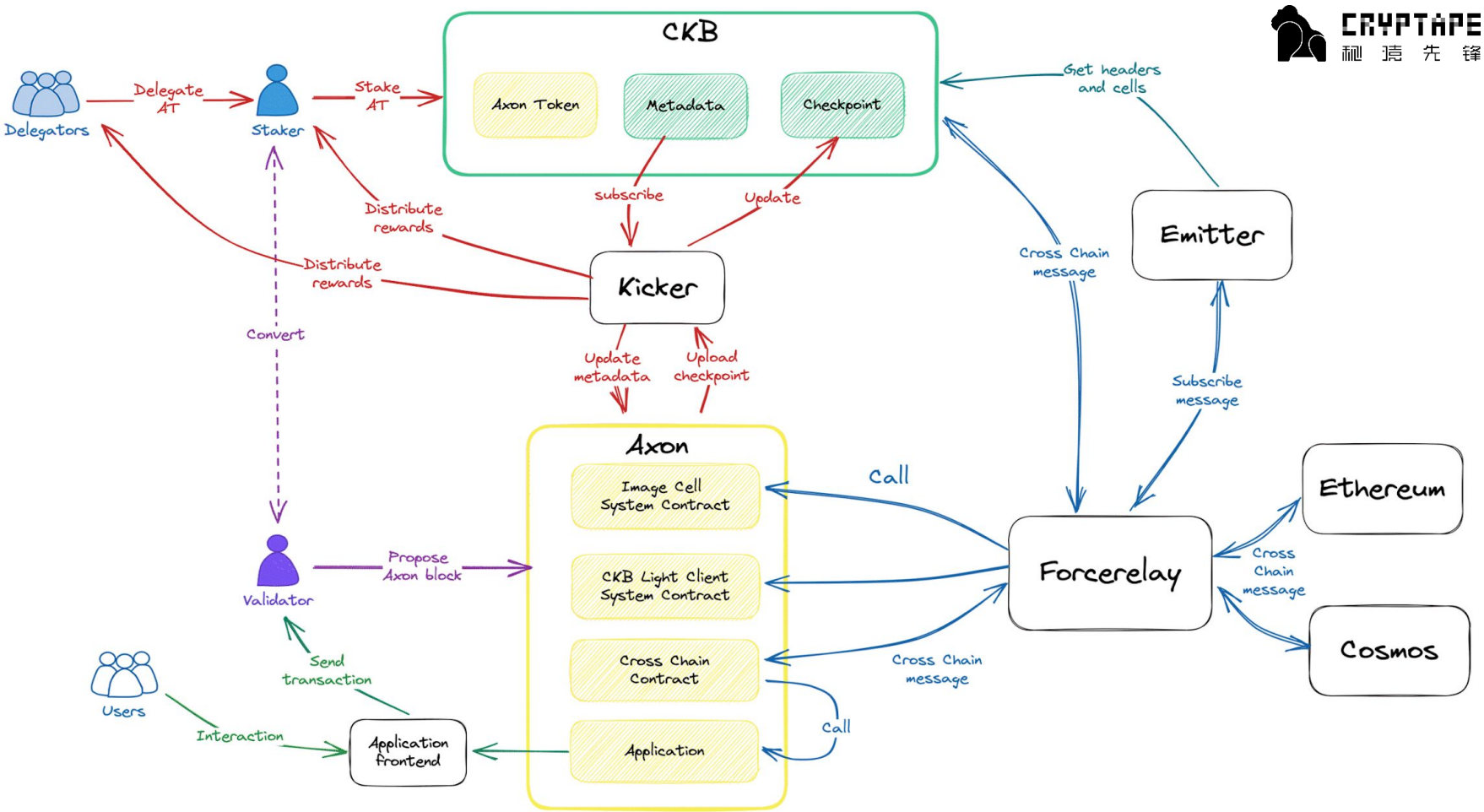
- IBC (Inter-Blockchain Communication)

  两个区块链之间传递任意数据的互操作性协议

CKB-VM：基于 RISC-V 指令集
+
ICSC 系统合约

Axon 可以执行任何
部署在 CKB 上的合约 (Rust)




CKB
Axon Token
Metadata
Checkpoint
Delegators
Delegate AT
Staker
Stake AT
Get headers and cells
subscribe
Update
Distribute rewards
Distribute rewards
Kicker
Emitter
Cross Chain message
Convert
Update metadata
Upload checkpoint
Subscribe message
Axon
Image Cell System Contract
CKB Light Client System Contract
Cross Chain Contract
Application
Call
Ethereum
Forcerelay
Cross Chain message
Cosmos
Validator
Propose Axon block
Send transaction
Cross Chain message
Users
Interaction
Application frontend
Call

Forcerelay

CRYPTAPE
神 猿 先 锋

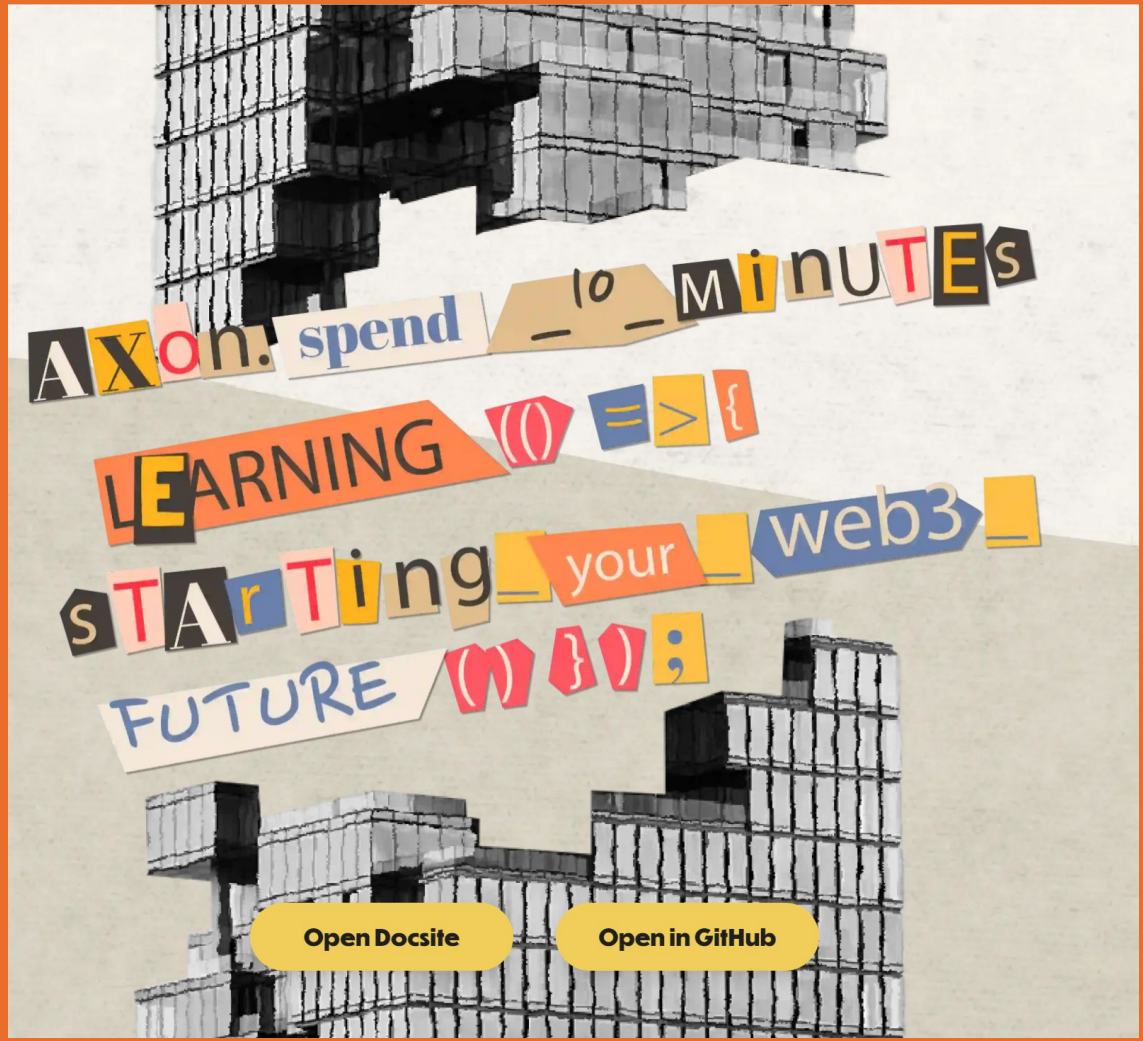
通过 IBC 协议，让 CKB 和基于 Axon 构建的区块链可以同以太坊和 Cosmos-SDK Chains 交互。

One more thing

官网：axonweb3.io

邮箱：axon@axonweb3.io

扫码即可进入官网



Thank you !