

DeFi on Solana · Week 4

Speaker script: every slide with its notes. Facts checked 11 to 14 Oct 2026; re-check anything marked "verify".

1

Week 4 of 6

4

Lending, borrowing & stablecoins

Task

Model a loan to liquidation in a spreadsheet, with a live Pyth price

Blog post

Understanding lending and borrowing in DeFi

DeFi on Solana · Week 4

Week 4 is credit. Accountants know secured lending, LTV and margin calls already. Today: what changes when the lender is a program, the borrower is anonymous, and the margin call is a bot.

Honest note for the class: this is the one task from the original curriculum that doesn't port cleanly. The original said "borrow and lend on Aave or Compound testnet". No Solana lender had a reliably working devnet market when this course was checked (11 October 2026). So everyone models it in a spreadsheet with a live price, builders can use a mainnet fork (Surfpool), and there is one optional real-money step, clearly flagged.

2

Today

1

A bank with no banker

2

The four numbers

3

Liquidation, step by step

4

Where prices come from

5

Stablecoins: the dollars of DeFi

Today

Five stops. Part 3 is the centrepiece: the liquidation walkthrough is what they'll model in the task. Part 4 links to week 5 (oracle manipulation is the Mango hack). Part 5 matters for Nigeria in particular.

3

A bank with no banker

Lenders, borrowers, and a program in the middle.

A bank with no banker

No notes. Let the slide speak.

4

No name, no credit score, no court

A bank lends on

- who you are
- your income and history
- the right to sue you

A lending program knows

- an address
- what that address locked up

So every loan is worth less than its collateral.

\$150 in → borrow at most \$105

No name, no credit score, no court

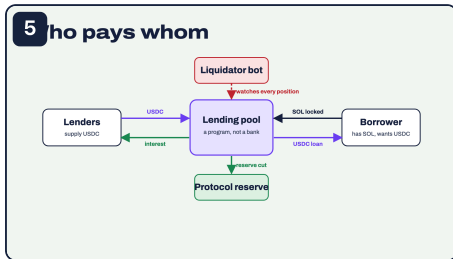
Why DeFi lending is over-collateralised.

A bank can lend unsecured because it knows who you are, can check your income, and can take you to court. A lending program on Solana knows only an address and what that address has locked up. If you walk away, there is no one to chase.

So the only protection is collateral worth more than the loan, plus a way to sell it before it drops below the loan's value. With a 70% maximum loan-to-value (example market), \$150 of SOL lets you borrow at most \$105.

Ask the room: why would anyone borrow less than they own? Answers: to get dollars without selling SOL (no taxable sale, keep the upside), to go leveraged long (borrow USDC, buy more SOL), to short (borrow SOL, sell it), or to farm a yield higher than the borrow rate.

Finance link: this is a secured margin loan with automated margin calls, closer to a broker's margin account than to a bank loan.



Who pays whom

Click 1: lenders supply USDC into a shared pool and get a receipt token (like a deposit certificate) that grows in value as interest accrues.

Click 2: a borrower deposits SOL as collateral. It is locked by the program, not handed to a person.

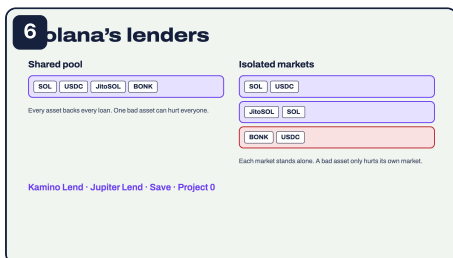
Click 3: the borrower takes USDC out of the pool, up to the max LTV.

Click 4: the borrower pays interest, continuously, into the pool. Lenders' receipts grow.

Click 5: the protocol keeps a slice of the interest (the reserve factor) as revenue and as a buffer against bad debt. Week 5 due diligence asks: where does the protocol's revenue come from? Here.

Click 6: liquidator bots watch every position on the chain, all the time. They are not employees; anyone can run one. They profit when a position becomes unsafe (part 3).

Difference from a bank: lenders here can see every loan and every piece of collateral. They cannot see who the borrower is. And there's no maturity date: loans stay open until repaid or liquidated.



Solana's lenders

The main Solana lenders as of October 2026: Kamino Lend, Jupiter Lend (launched August 2025), Save (formerly Solend), and Project 0 (formerly marginfi). Same mechanics as Aave and Compound on Ethereum: collateral, LTV, liquidation.

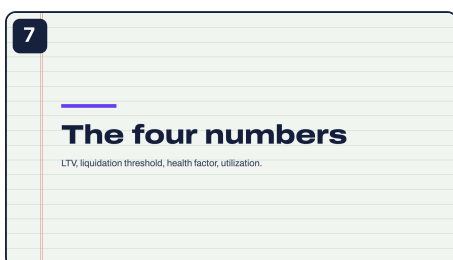
Two designs:

- Shared pool: all assets sit in one pool and every deposit backs every loan. Efficient, but a bad or manipulated asset can create bad debt that the whole pool absorbs. This is exactly what the Mango Markets attack exploited (week 5).
- Isolated markets: each market is its own pair or small group. A failure in the BONK/USDC market cannot touch the SOL/USDC market. Safer, more fragmented.

Most modern lenders mix the two. Kamino, for example, runs a main market and separate isolated ones.

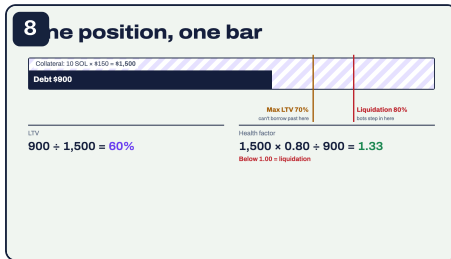
marginfi's team relaunched as Project 0 (P0), announced 11 Sep 2025, on the same lending program. The old marginfi app, SDK and API were shut down in mid-2026; the app is now app.0.xyz. It was a rename, not a sale.

TVL rankings move weekly. If you quote one, open DefiLlama and say the date.



The four numbers

No notes. Let the slide speak.



One position, one bar

Example market (not a real one): max LTV 70%, liquidation threshold 80%.

Real markets set these per asset; always look them up.

Click 1: collateral. 10 SOL at \$150 = \$1,500.

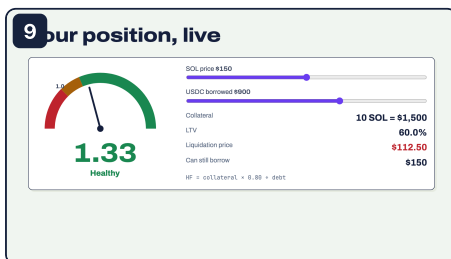
Click 2: debt. You borrowed 900 USDC. LTV = debt $\div$ collateral = $900 \div 1,500 = 60\%$.

Click 3: max LTV, 70%. The program won't let you borrow beyond \$1,050 against this collateral. That's the "how much can I borrow" number.

Click 4: liquidation threshold, 80%. If your LTV ever reaches 80% (because SOL falls or interest piles up), liquidators may repay part of your debt and take your collateral at a discount. The gap between 70% and 80% is your safety buffer.

Click 5: health factor. HF = collateral $\times$ liquidation threshold $\div$ debt = $1,500 \times 0.8 \div 900 = 1.33$. Above 1 is safe. At 1.00 exactly, you are on the line. Below 1, liquidatable. Kamino, Save and others show some version of this number in the app.

Students mix up max LTV and liquidation threshold constantly. One is a ceiling for borrowing, the other a trigger for selling. Draw this bar on the board.



Your position, live

Live widget, same example market (70% / 80%), 10 SOL collateral.

Do these in order:

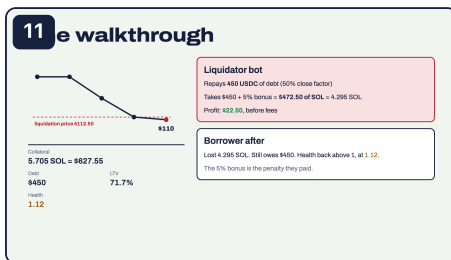
- Start: \$150, \$900 borrowed. HF 1.33, liquidation price \$112.50.
Liquidation price = debt $\div$ (SOL amount $\times$ threshold) = $900 \div (10 \times 0.8) = \$112.50$. This is the single most useful number for a borrower.
- Drag price to \$130: HF 1.16, "one bad candle away".
- Drag to \$110: HF 0.98. Liquidatable. This is the handbook's walkthrough; the next slide replays it.
- Reset to \$150, push the loan to \$1,050: you're at max LTV. Notice "can still borrow" hits \$0.
- Push it to \$1,200: the program wouldn't allow this, but the widget does, so they can see HF drop below 1 instantly.

Interest is not modelled here. In reality debt grows every slot, so an untouched position drifts towards liquidation even if prices stay flat.



Liquidation

No notes. Let the slide speak.



The walkthrough

This is the handbook's example. Example market: liquidation threshold 80%, close factor 50%, liquidation bonus 5%. Real numbers vary per protocol and asset.

Click 1: deposit 10 SOL at \$150 (\$1,500). Borrow 900 USDC. LTV 60%, HF 1.33.

Click 2: SOL falls to \$130. Collateral \$1,300. LTV 69.2%, HF 1.16.

Click 3: SOL at \$112.50. LTV exactly 80%. HF 1.00. On the line.

Click 4: SOL at \$110. Collateral \$1,100. $LTV = 900 \div 1,100 \approx 82\%$. HF 0.98. The bot's alarm goes off.

Click 5: the liquidator repays 450 USDC of the borrower's debt (the close factor caps it at 50% per liquidation in this example).

Click 6: in return, the program gives the liquidator \$450 of SOL plus a 5% bonus: \$472.50 of SOL, which at \$110 is 4.295 SOL. Their profit is \$22.50, minus transaction fees and any slippage when they sell the SOL. Bots compete for this, so it happens within seconds.

Click 7: the borrower now has 5.705 SOL (\$627.55) against \$450 debt. $HF = 627.55 \times 0.8 \div 450 = 1.12$. Safe again, but they paid a \$22.50 penalty and lost almost half their SOL.

Why this is good for lenders: the debt is repaid while the collateral is still worth more than it. If prices crash too fast for liquidators (or the oracle lags), collateral can end up worth less than the debt. That's bad debt, and lenders absorb it.

Ask the teacher-prep question out loud: compare 82% to a real market's liquidation threshold for SOL on Kamino or Jupiter Lend. Look it up live.



Flash loans: a million, for one transaction

Liquidators often don't have 450 USDC lying around. Flash loans solve that.

Click 1: borrow any amount the pool has, with no collateral.

Click 2: use it. Typical uses: repay someone's debt in a liquidation and receive their discounted collateral; arbitrage between two pools; refinance a position from one lender to another.

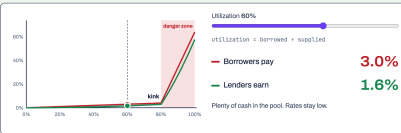
Click 3: repay the loan plus a small fee, in the same transaction. If you do, it lands, and you keep the profit.

Click 4: the other ending. If the loan isn't repaid by the end of the transaction, the program refuses to finish, and the whole transaction is undone, as if nothing happened. The only cost is the network fee.

On Solana the "same transaction" rule is enforced by instructions inside one transaction (the lender checks that a repay instruction comes later in the same transaction).

The dark side, for week 5: flash loans also let an attacker borrow huge sums for a few hundred milliseconds to manipulate a thin market's price. The loan isn't the bug; the bug is a protocol trusting a price that one transaction can move.

13 What borrowing costs



What borrowing costs

Live widget. Example parameters (not a real market): borrow rate rises from 0% to 4% as utilization goes to 80% (the kink), then from 4% to 64% between 80% and 100%. Reserve factor 10%.

Utilization = borrowed ÷ supplied. It's the share of the pool's cash that's lent out.

1. At 60%: borrowers pay 3%, lenders earn 1.6%. Lenders earn less than borrowers pay because (a) only 60% of their money is lent out, and (b) the protocol keeps 10%. Lender rate = borrow rate × utilization × (1 – reserve factor).
2. Drag to 80%: the kink. 4% borrow, about 2.9% supply.
3. Drag to 95%: borrowing jumps to about 49%. Why so steep? At 100% utilization there is no cash left, so lenders cannot withdraw. The steep slope pulls in new deposits and pushes borrowers to repay, defending withdrawals.

Finance link: this is a money market with an administered rate curve instead of a central bank. Ask: what happens in a panic, when everyone wants to withdraw at once? (Utilization spikes, rates explode, and some lenders wait. Liquidity risk, not credit risk.)

14 Quiz: 10 SOL collateral, 900 USDC debt, liquidation threshold 80%. At what SOL price can you be liquidated?

☐ A \$90.00
☐ B \$112.50
☐ C \$128.57
☐ D \$150.00

Quiz: 10 SOL collateral, 900 USDC debt, liquidation threshold 80%. At what SOL price can you be liquidated?

C (\$128.57) is the max LTV price ($900 \div (10 \times 0.7)$): the point where you can't borrow more, not where you get liquidated. A good wrong answer to discuss.

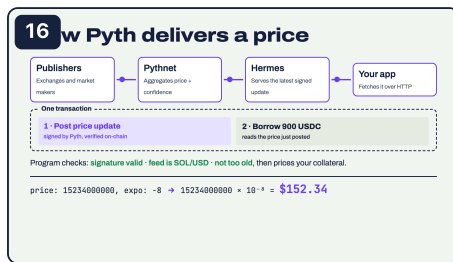
15

Where prices come from

A lender is only as safe as its price feed.

Where prices come from

No notes. Let the slide speak.



How Pyth delivers a price

A Solana program cannot look at Binance. It only sees accounts passed into its transaction. So prices must be brought on-chain by an oracle. Pyth is the main one on Solana (Switchboard is the other).

Click 1: publishers. Exchanges, market makers and trading firms publish their own prices for SOL/USD.

Click 2: Pythnet, Pyth's own appchain, aggregates them into one price plus a confidence interval, continuously.

Click 3: Hermes is Pyth's web service. It serves the latest price update, signed so that the Pyth program on Solana can verify it.

Click 4: your app (the lending UI) fetches the latest update from Hermes (apps use an API key since 26 Aug 2026).

Click 5: this is the "pull" part. The app puts two instructions in one transaction: first post the signed price update, then borrow. The price arrives with the user's action, so it's fresh.

Click 6: the lending program checks the update is genuine (signature), is the right feed (SOL/USD, not some other account), and isn't too old. Then it values your collateral.

Click 7: decoding. Prices come as an integer plus an exponent. $15234000000 \times 10^{-8} = \$152.34$. The confidence interval is encoded the same way.

Compare with Chainlink on Ethereum, which is mostly "push": the oracle writes prices on a schedule and contracts read the last value.

Why it matters for week 5: check 6 is everything. A program that accepts a price account without verifying what it is, or that trusts a price from a thin market, can be drained. Mango (2022) and Loopscale (2025) were pricing failures.

SOL/USD feed id:

ef0d8b6fda2ceba41da15d4095d1da392a0d2f8ed0c6c7bc0f4cfac8c280b56d

17

e the live price

Users

insights.pyth.network

Search SOL/USD. Read the price, the confidence band and when it last updated.

Builders

npm run w4:price

Reads Pyth's SOL/USD account on Solana: price * 10^expo, after checking who owns the account.

Hermes now needs a key

Since 29 Aug 2020 the public Hermes URL returns 401. Apps use a key; you don't need one for class.

Write down the price and the time you looked. That's your spreadsheet's starting row.

See the live price

Checked 14 Oct 2026: Pyth's Hermes service now requires an API key (Authorization: Bearer). The old browser URL `hermes.pyth.network/v2/updates/price/latest?ids[]=...` returns 401 unauthorized, so the original "paste the URL" exercise no longer works. Source: docs.pyth.network/price-feeds/core/upgrade/preparing.

Users: open `insights.pyth.network` and search SOL/USD. It shows the aggregate price, the confidence interval (how much the publishers disagree) and the last update. Ask: why does a lender care about the confidence band? (A wide band means the price is uncertain; careful lenders treat that as risk.)

Builders: `cd builders/ts && npm run w4:price`. It reads the SOL/USD price account `7UVimffxr9ow1uXYxsr4LHAcV58mLzhmwaeKvJ1pJLiE` (owned by the Pyth Receiver program) straight from Solana mainnet, no key needed. The raw numbers come as an integer price and an exponent: `dollars = price × 10^expo`. The script also checks the account's owner before trusting it, which is the week 5 lesson (Wormhole, Cashio) in one line. With a free Pyth key (`pythdata.app/signup`) it also fetches the signed Hermes update.

The pull model on the previous slide still holds: apps fetch a signed update and post it in the same transaction. Only the public, keyless access went away.

The other big oracle on Solana is Switchboard. A common pattern is pricing something in dollars but paying in SOL: a bounty app can ask for "\$100 of SOL", read the SOL/USD feed at funding time, and lock that much SOL in escrow. Ask: what happens to the prize's dollar value if SOL drops 30% before the winner is paid? (It drops too, unless the app re-prices or holds a stablecoin.)

This price is the starting row of the spreadsheet task. Course rule 3: write down the time.

18

Stablecoins

The dollars of DeFi.

Stablecoins

No notes. Let the slide speak.



Three ways to hold \$1

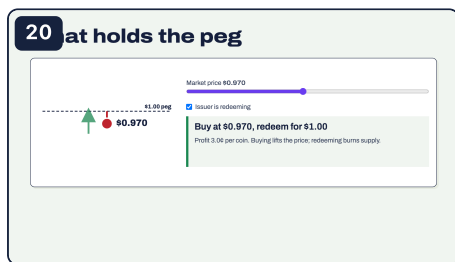
The bars show what backs one coin. Dashed line = \$1 of backing.

Click 1, fiat-backed: a company holds cash and short-term government bills, and redeems 1 coin for \$1 (for verified customers). USDC (Circle), USDT (Tether), PYUSD (issued by Paxos for PayPal), USDG (Paxos, from Singapore). This is almost all of Solana's stablecoin supply. Risks: the issuer, its banks, the quality of reserves, and the issuer's power to freeze coins (they can, on Solana too).

Click 2, crypto-backed: DAI from MakerDAO (rebranded to Sky; the new coin is USDS). Users lock more than \$1 of crypto per coin minted (the bar goes above the line), and vaults are liquidated if they slip, exactly like today's lending. Teach it as history and contrast: Solana has no big native coin of this kind.

Click 3, algorithmic: UST on Terra. Backed by nothing but the market value of a sister token (LUNA): you could always burn \$1 of LUNA to mint 1 UST. When confidence broke in May 2022, people redeemed UST for LUNA, LUNA's price collapsed, which made UST less credible, and both spiralled to near zero. Tens of billions of dollars of value disappeared within days. The bar is below the line on purpose.

Solana numbers: stablecoin supply on Solana hit a reported record of \$17.3B on 25 September 2026, mostly USDC (Solana Compass). Quote with the date.



What holds the peg

Live widget, for fiat-backed coins.

1. Price \$0.97: arbitrageurs buy at 97 cents and redeem with the issuer for \$1. Buying lifts the market price; redeeming removes coins. The green arrow is that pull.
2. Drag above \$1: mint new coins at \$1 from the issuer and sell at \$1.02. Selling pushes the price down.
3. Untick "issuer is redeeming". The pull disappears. Nothing forces the price back except belief.

Real example: in March 2023 Circle said about \$3.3 billion of USDC reserves were stuck at Silicon Valley Bank, which failed on a Friday. Banks were closed for the weekend, redemptions couldn't settle, and USDC traded down to roughly 87 to 88 cents. It came back to \$1 once US authorities said SVB depositors would be made whole. Peg mechanisms depend on the plumbing behind them.

The curriculum's line: don't assume a stablecoin is the same as insured bank money just because it targets \$1. Compare: who owes you, can you redeem, is there deposit insurance (no), can it be frozen (yes).

21 Why it matters here

Payments

Settles in under a second for a fraction of a cent, any day of the week.

Remittance

Wallet to wallet across borders, no correspondent bank.

Savings

A dollar balance without a dollar bank account.

The catch

Off-ramps, local rules, issuer freezes, and scams.

Why it matters here

Bring it home. The handbook asks: why did PayPal and Paxos choose Solana, and what does this mean for payments in Nigeria?

- Payments: Solana fees are fractions of a cent and blocks come every ~400ms, so a stablecoin transfer is cheap enough for small everyday payments. That's the pitch PayPal (PYUSD) and Paxos (USDG) bought into.
- Savings: for people facing a weak local currency, a dollar stablecoin is a way to hold dollars without a domiciliary account.
- Remittance: send from a wallet abroad to a wallet at home, without correspondent banks.
- The catch: you still need an off-ramp to naira, local regulation applies (cover SEC Nigeria's current stance on digital assets in week 5), issuers can freeze coins, and "send me USDT first" scams are everywhere.

Don't quote adoption statistics without a dated source. Ask students who has used USDT or USDC already, and for what. Their answers are better than any statistic.

22

trades at \$0.96 and Circle is redeeming normally. What pulls it back to \$1?

A Circle prints more USDC

B Traders buy at \$0.96 and redeem for \$1

C A Solana validator vote

D Nothing, it stays there

Quiz: USDC trades at \$0.96 and Circle is redeeming normally. What pulls it back to \$1?

D is correct if redemption stops (the SVB weekend). That's the follow-up question.

23 Task: take a loan to liquidation

Users

Simulate it (recommended)

1. Get the live SOL price (insights.pyth.network)
2. Spreadsheet: deposit, borrow, then drop the price in steps
3. Each row, LTV, health factor
4. Mark the liquidation price and the liquidator's profit

Optional, real money

Already hold USDC? Supply \$5 to a lender and watch interest accrue. Supply only. Never borrow.

Builders

Mainnet fork

1. `surfpool start`: real Kamino and Jupiter Lend accounts, locally
2. Fund a wallet with cheatcodes
3. Deposit and borrow through the SDK
4. Move time and price, watch health

Task: take a loan to liquidation

Checked 14 Oct 2026: no major Solana lender offers a public devnet UI. Kamino's lending program is deployed on devnet at its mainnet ID (KLend2g3cP87ffoy8q1mQqGKjrxjC8boSyAYavgmjD), so builders can create their own devnet market with `klend-sdk` (kamino-manager create-market). The other route is a Surfpool mainnet fork. Users do the spreadsheet (materials/week-4-lending-model.xlsx).

Everyone, simulate (recommended): spreadsheet with columns price, collateral value, debt, LTV, HF, status. Start at the live Pyth price ($\text{price} \times 10^{\text{expo}}$), borrow at 60% LTV, then step the price down 5% per row until $\text{HF} < 1$. Then add the liquidation row: repay 50%, seize $\text{debt} \times 1.05$ of collateral. Use the real liquidation threshold for SOL from Kamino or Jupiter Lend's UI (look it up, write down where and when).

Builders, mainnet fork: `surfpool start` runs a local validator on port 8899 that fetches real mainnet accounts as you touch them, so Kamino and Jupiter Lend behave as they do live. Use cheatcodes to fund a wallet and move time, then deposit and borrow through the SDK. Reference repo: `adpthegreat/bypassing-kamino-restrictions`.

Optional real money: mentees who already hold USDC can supply \$5 to a lender and watch interest accrue. Supply only, no borrowing. Say clearly, out loud, that it is real money and optional. Nobody should buy crypto for this.

24

Blog post**"Understanding lending and borrowing in DeFi"**

Include your liquidation walkthrough, and one paragraph on which stablecoin you'd trust, and why.

Blog post

The curriculum's week 4 report also asks for: a hypothetical lending calculation, a comparison of stablecoin designs, a case study of a documented incident (USDC and SVB, or UST, work well), and the difference between credit risk, market risk, liquidity risk, collateral risk and smart-contract risk. The blog post can be the short version.

25

Teacher prep

For you, before class. Skip this live.

Teacher prep

No notes. Let the slide speak.

26 You, before class

1. **Learn the four numbers.** LTV, liquidation threshold, health factor, utilization. Then the linked rate curve.
2. **Walk one liquidation by hand.** 10 SOL at \$150, borrow 900 (60%). SOL to \$110: LTV = 82%.
3. **Look up SOL's real liquidation threshold** on Kamino or Jupiter Lend. Who gets liquidated, and how does the liquidator profit?
4. **Read Pyth's Solana pull-integration guide.** Week 6 depends on it.
5. **Know the stablecoin types,** and which have real depth on Solana.

You, before class

From the handbook's "You, before class" lane for week 4.

The liquidation by hand: deposit 10 SOL at \$150 (\$1,500), borrow 900 USDC (60% LTV). SOL drops to \$110, so LTV is $900 / 1,100 \approx 82\%$. Compare that to the market's real liquidation threshold for SOL. The slides use an example 80% threshold, 50% close factor and 5% bonus; replace them with the real market's parameters if you have them.

Stablecoin types: fiat-backed (USDC, USDT, PYUSD, USDG from Paxos Singapore), crypto-backed (DAI, now Sky), synthetic and yield-bearing. Know which ones actually have depth on Solana (USDC dominates).

Pyth: read the Solana pull-integration guide, and the Pyth + Anchor devnet tutorial if you have builders.

27 Verify before class**Any devnet lending market**

No lender has a public devnet UI. Builders: create a Kamino devnet market with bland sol, or fork mainnet with Burfpool.

marginfi is now Project 0

Same team and program, renamed Sep 2025. Old app shut in mid-2026; use app.0.xyz.

Numbers with dates

Stablecoin supply (\$17.3B, 25 Sep 2026) and TVL, move weekly. Re-check on DefiLlama the day you teach.

Builders' fork

`burfpool_start` - port 8888 - cheatcodes for balances and time

Verify before class

All from the handbook's verify list, checked 11 October 2026. Trackers disagree, sometimes by 2x. Use DefiLlama directly on the day and say the date out loud.