I'm not a robot

This guide will take you through the basics of how to migrate to Nimbus from another client.See here for advanced options.Please take your time to get this right.Don't hesitate to reach out to us in the #helpdesk channel of our discord if you come across a stumbling block.We are more than happy to help guide you through the migration process.Given what's at stake, there is no such thing as a stupid question. WarningThe most important takeaway is that you ensure that two clients will never validate with the same keys at the same time.In other words, you must ensure that your original client is stopped, and no longer validating, before importing your keys into Nimbus.Steps1. Sync the Nimbus beacon nodeNo matter which client you are migrating over from, the first step is to sync the Nimbus beacon node.The easiest and fastest way to do this is to follow the beacon node quick start guide and perform a trusted node sync from the source client.Once your Nimbus beacon node has synced and you're satisfied that it's working, move to Step 2.TipYou can keep track of your syncing progress with the following command:curl -X GET for an "is_syncing":false in the response to confirm that your node has synced.2. Stop your existing client and export your slashing protection historyAs part of the migration process, you need to stop your existing client and export its slashing protection database.PrysmLighthouseTekuNimbus Stop and disable the Prysm validator client (you can also stop the Prysm beacon node if you wish).If you're using systemd and your service is called prysmvalidator, run the following commands to stop and disable the service:sudo systemctl stop prysmvalidator.servicesudo systemctl disable prysmvalidator.serviceIt is important that you disable the Prysm validator as well as stopping it, to prevent it from starting up again on reboot.2. Export slashing protection historyRun the following to export your Prysm validator's slashing protection history:prysm.sh validator slashing-protection-history export \--datadir=/your/prysm/wallet \--slashing-protection-export-dir=/path/to/export_dirYou will then find the slashing-protection.json file in your specified /path/to/export_dir folder. The validator client needs to be stopped in order to export, to guarantee that the data exported is up to date.If you're using systemd and your service is called lighthousevalidator, run the following command to stop and disable the service:sudo systemctl stop lighthousevalidatorsudo systemctl disable lighthousevalidatorYou may also wish to stop the beacon node:sudo systemctl stop lighthousebeaconsudo systemctl disable lighthousebeaconIt is important that you disable the service as well as stopping it, to prevent it from starting up again on reboot.2. Export slashing protection historyYou can export Lighthouse's database with this command:lighthouse account validator slashing-protection export slashing-protection.jsonThis will export your history in the correct format to slashing-protection.json. If you're using systemd and your service is called teku, run the following command to stop and disable the service:sudo systemctl stop tekusudo systemctl disable tekuIt is important that you disable the service as well as stopping it, to prevent it from starting up again on reboot.2. Export slashing protection historyYou can export Teku's database with this command:teku slashing-protection export --data-path=/home/me/me_node --to=/home/slash/slashing-protection.jsonWhere:--data-path specifies the location of the Teku data directory.--to specifies the file to export the slashing-protection data to (in this case /home/slash/slashing-protection.json). Once your Nimbus beacon node on your new setup has synced and you're satisfied that it's working, stop and disable the Nimbus validator client on your current setup.If you're using systemd and your service is called nimbus-eth2-mainnet, run the following commands to stop and disable the service:sudo systemctl stop nimbus-eth2-mainnet.servicesudo systemctl disable nimbus-eth2-mainnet.serviceIt is important that you disable the service as well as stopping it, to prevent it from starting up again on reboot.2. Export slashing protection historyRun the following to export your Nimbus validator's slashing protection history:build/nimbus_beacon_node slashingdb export slashing-protection.jsonThis will export your history in the correct format to slashing-protection.json.TipTo be extra sure that your validator has stopped, wait a few epochs and confirm that your validator has stopped attesting (check its recent history on beaconcha.in).Only after that, continue with the next step of this guide.3. Import your validator key(s) into NimbusTo import your validator key(s), follow the instructions in our validator guide.TipTo check that your key(s) has been successfully imported, look for a file named after your public key in build/data/shared_mainet_0/secrets/.If you run into an error at this stage, it's probably because the wrong permissions have been set on either a folder or file.See here for how to fix this.4. Import your slashing protection historyTo import the slashing protection history you exported in step 2, from the nimbus-eth2 directory run:build/nimbus_beacon_node slashingdb import path/to/export_dir/slashing-protection.jsonReplacing /path/to/export_dir with the file/directory you specified when you exported your slashing protection history.TipAdditional slashing protection information can be safely added to slashing protection databases.5. Start the Nimbus validatorFollow the instructions in our validator guide to start your validator using our pre-built binaries.If you prefer to use Docker, see our Docker guide.For a quick guide on how to set up a systemd service, see our systemd guide.Final thoughtsIf you are unsure of the safety of a step, please get in touch with us directly on Discord.Additionally, we recommend testing the migration works correctly on a testnet before going ahead on mainnet. New to Reddit? Create your account and connect with a world of communities. Staking is a necessary piece of Ethereum. Fundamentally, staking is the mechanism which enables Ethereum (and other PoS networks) to reach consensus on which blocks to produce in a decentralized manner (read Staking is data validation, not investment). Without consensus, there is no peer-to-peer coordination at scale and there is no web3: this critical function underlies the entire wave of permissionless web3 innovation.Through staking, Ethereum offers economic incentives to network participants to contribute to network coordination and security. By staking their ETH and operating validators, users contribute to Ethereum consensus mechanism security and are rewarded for it by earning rewards.Slashing plays a fundamental role in maintaining the security and integrity of Ethereums Proof-of-Stake (PoS) mechanism. It is designed to deter validators from engaging in dishonest behavior or breaching their obligations. The concept might sound stern, but it is essential to preserving a healthy ecosystem: you earn rewards if you contribute to Ethereum consensus mechanism, and you are penalized if you misbehave.In this article, we delve deep into what slashing means, its relevance in the staking ecosystem, and the mechanisms in place to prevent it.Slashing on EthereumIn Ethereums PoS ecosystem, slashing refers to the process of penalizing a validator for misbehaving. Validators, entrusted with the tasks of attesting and proposing blocks, are essential for maintaining the networks smooth operations. If a validator is found to behave maliciously or dishonestly, it may be slashed, which means that a portion of its deposit is destroyed by the network as a punishment. As such, slashing incentivises validators to perform their duties correctly, thereby promoting the networks best interests.There are three ways a validator can be slashed, all of which amount to a dishonest proposal or attestation of blocks:By proposing and signing two different blocks for the same slotBy attesting to a block that "surrounds" another one (effectively changing history)By attesting to two candidates for the same block ("double voting")Historically, slashing occurrences have been quite rare. As of February 2024, 414 validators have been slashed out of approximately 1,174,000 deposited validators (916,000 are active to date). This accounts for less than 0.04% of all active validators. As far as we know, all these slashing events are due to operational errors rather than intentional misconduct.Since its launch in December 2020, Consensys Staking validators have never been slashed.Slashing PenaltiesA validator incurs a slashing penalty when committing a slashable offense, pushing them into a slashed state on Ethereum.Slashing penalties are the same for all slashable offences, and have multiple components:First, there is an immediate initial penalty of ~1ETH, or more precisely 1/32 of the validators effective balance, once the offence is identified.Once slashed, the validators status is set to slashed_exiting. For about 36 days, the validator is removed from the active validation set and is placed in the exit queue. During this period, the validator not only stops earning new rewards but also incurs a penalty of about 8,000 GWei (0,000008 ETH) for every epoch that it misses performing its duties (ie. every 6.4 minutes). This leads to an additional penalty of ~0.07ETHFinally, a special penalty may be applied depending on the number of validators who committed a slashable offense a few days before and after: the more validators are slashed simultaneously, the bigger the penalty per validator will be. This is meant to prevent coordinated attacks on the Ethereum network.In extreme cases, the slashing penalty could wipe all of a validators staked ETH. However, historically, most slashing penalties have been around 1ETH.Mitigating Slashing Risks with Consensys StakingA majority of slashing events occur unintentionally when two different validators clients use the same validator key. Therefore, our anti-slashing strategy at Consensys Staking revolves around the simple rule that we do NOT deploy the same validator key to multiple locations or multiple validators.To ensure we never break this simple rule, we employ several practices:Each validator key is generated using a distinct seed phrase, ensuring each key is unique.A robust check mechanism is in place to detect any duplicate keys before the validator public keys are made available.We also use web3 signer, an open-source remote signer built by Consensys, which maintains a record of all recent signatures made by a validator and checks each signature against potential slashing conditions.Beyond these measures, Consensys Staking is dedicated to maintaining robust security practices. Leveraging multiple layers of security, including rigorously reviewed access permissions and minimum access policies, we continually work to mitigate potential attacks. Consensys is ISO 27001:2022 certified, a security assurance covering the security of our people, technology and processes. Consensys Staking was also granted SOC 2 Type II certification, reinforcing our position as a secure and reliable partner in your staking journey.Stake ETH Your Way, with Consensys Staking Slashing is a scary word. But what exactly is it, how can it happen and how worried should you be? What is Slashing?Slashing is a term used to describe the response of the Ethereum network to a validator acting against the rules of the network. Validators perform a number of duties (e.g. attestations and proposing blocks).If someone wanted to attack the Ethereum network they could propose multiple blocks or attest to multiple conflicting blocks. To disincentivize attacks on the network, in a Proof of Stake (PoS) system, validators have something at stake, which is currently 32 ETH per validator. When a validator breaks the rules of the network, two things will happen:The validator has some amount of ETH taken from that initial 32 ETH staked balance.The validator is force exited and removed from the validator pool.The amount of ETH taken as a penalty varies on the state of the network. If a small number of validators are slashed simultaneously, then a rough estimate of the slashing penalty is 1 or 2 ETH. In an incredibly rare Black Swan event, when a large portion of the network is simultaneously offline or breaking the rules (e.g. in a coordinated attack) then the slashing penalty can be up to and including 100% of the stake.When your validator is force exited and the stake is withdrawn you are able to re-stake your remaining ETH (if you still have the 32 required), after going through both the exit queue and activation queue again.Slashing Tracker Slashings | Ethereum Mainnet You cant perform that action at this time. A technical handbook on Ethereum's move to proof of stake and beyondEdition 0.3: Capella [WIP] Created by Ben Edgington. Licensed under CC BY-SA 4.0. Published 2025-09-01 08:19 UTC. Commit d013cb9. Ethereum 2.0 has introduced a new mechanism, called "slashing," designed to penalize malicious validators.However, some common user mistakes may result in slashingeven if there was no ill intent.An Ethereum protocol developer at Prysmatic Labs explained how to avoid such mistakes.With the launch of Ethereum (ETH) 2.0 Phase 0 on December 1, users were introduced to a bunch of new features and nuances that the proof-of-stake (PoS) consensus mechanism has brought with it. One of them is slashinga network protection mechanism that punishes validators if they dont fulfill their task correctly.Raul Jordan, Ethereum protocol developer at Prysmatic Labs, published a comprehensive blog post on November 30, explaining how to avoid getting slashed.What is slashing?On the simplest level, slashing is a form of law enforcement on Ethereum 2.0.This mechanism is designed to detect and suppress any validators activity that can be harmful to the network. This includes, for example, proposing or attesting to two different conflicting blocks in the same slot or casting a vote which surrounds or is surrounded by a previous one.When an active validator gets slashed, the system begins to gradually destroy portions of its ETH stake (essentially burning money) over the course of 36 days, after which the guilty actor is booted from Ethereum 2.0s Beacon Chain. Notably, slashing is irreversible, meaning that users will have to generate new validator keys and deposit new stakes if they want to continue validating after being slashed.However, some common user mistakes can also result in slashingeven if there was no ill intent.How to avoid being slashed?According to Jordan, one of the biggest mistakes that can lead to slashing is when a user inputs the same validating keys into two or more servers (keeping one of them as a backup, for example).This is the EASIEST way to get yourself slashed. If your failover system has a false positive your first node is down, you can find yourself in a situation where you commit slashable offenses, Jordan explained.To avoid this, users should never run identical validating keys in two or more places at the same time. Especially since most of the slashing protection instruments wont help in this specific case.I have just talked to the slashed validator and we found the issue at hand. They were running another instance of their validator. Let this be a warning to you:Do NOT run your validators in more than one place and validator instance. phil.eth (@phil_eth) December 2, 2020Another common mistake is when a user migrates his validator to a different machine or ETH 2.0 clientbut forgets to also relocate the slashing protection historya database that contains a local signing history.This database ensures the validator does not sign a message that would be considered a slashable message according to its own history; more simply, the validator sees the database as its sole source of truth when deciding if a message should be signed, Jordan noted, adding, This approach ensures the single validator does not perform duplicative actions.He also explained that slashing protection history is currently one of the most effective and easiest ways to safeguard your validators from being slashed. Thus, users should always migrate it as well when switching to new hardware or softwareand avoid losing it altogether.Another way to easily get slashed is by using a containerized or cloud environment that doesnt have persistent volumes.You need to setup persistent volumes for your validators, such that if a pod or container is restart, (sic) the slashing protection history will not be erased, Jordan added.Finally, the worstalthough very unlikelyscenario involves bugs in implementations. To avoid them, it is critical that users understand how to set up, configure, upgrade, and troubleshoot any installed software.Ultimately, slashing protection history is the first and foremost line of defense that should protect honest validators in most scenarios. But its best to double-check.Start every day with the top news stories right now, plus original features, a podcast, videos and more.

How make resin. How make resin art. How to make forms for epoxy resin. Cara membuat resin foto. How to get resinite fibers in factory simulator. How to make resin case.