Data Management

Managing the data on your Sui Full node is critical to ensuring a healthy Sui network. This topic provides a high-level description of data management on Sui Full nodes that you can use to optimize your Full node configuration. For more information about Sui Full nodes, such as pruning policies and archival settings, see Run a Sui Full Node .

The minimal version of a Sui Full node executes all of the transactions Sui validators commit. Sui Full nodes also orchestrate the submitting of new transactions to the system:

The preceding image shows how data flows through a Full node:

A Sui Full Node stores multiple categories of data in its permanent store.

The per-epoch Sui store is beyond the scope of this topic. Sui uses the per-epoch store (resets at the start of each epoch) internally for authority and consensus operations.

Sui Full nodes store the following types of data:

Sui Full nodes support more than 40 RPC types that includes the following categories:

Refer to Access Sui Data for an overview of options to access Sui network data.

A Sui archive instance stores the full Sui transaction history since genesis in a database agnostic format. This includes information about transactions (with client authentication), effects, events, and checkpoints. As such, archival storage can be used for data auditing and for replaying historic transactions.

The current archival storage format doesn't include historic object versions.

As a Full node operator, you can enable archival fallback for your Full node by specifying the URL to upload archival data. Currently, Mysten Labs manages a Sui archive and stores it in AWS S3. To ensure a healthy network, we encourage the Sui community to set up additional archives to ensure archival data availability across the network. In a typical configuration, an archive trails behind the latest checkpoint by approximately 10 minutes.

A Full Node that starts from scratch can replay (and thus re-verify) transactions that occurred since Sui genesis from the given archive via configuring Archival Fallback in the fullnode.yaml configuration file to point to the S3 bucket that stores the archive.

A Sui Full node that fails to retrieve checkpoints from its peers via state sync protocol falls back to downloading the missing checkpoints from its pre-configured archive. This fallback enables a Full node to catch up with the rest of the system regardless of the pruning policies of its peers.

As described previously, sustainable disk usage requires Sui Full nodes to prune the information about historic object versions as well as historic transactions with the corresponding effects and events, including old checkpoint data.

Both transaction and object pruners run in the background. The logical deletion of entries from RocksDB ultimately triggers the physical compaction of data on disk, which is governed by RocksDB background jobs: the pruning effect on disk usage is not immediate and might take multiple days.

To learn more about object pruning policies, see Object pruning . You can configure the pruner in two modes:

Keeping some history (5 epochs) provides better RPC performance as it allows more lookups to happen from local disk rather than from remote storage.

To learn more about transaction pruning policies, see Transaction pruning . To configure transaction pruning, specify the num-epochs-to-retain-for-checkpoints: X config option. The checkpoints, including their transactions, effects and events are pruned up to X epochs ago. We suggest setting transaction pruning to 2 epochs.

In case your Full node is configured to upload committed information to an archive, you should ensure that pruning doesn't occur until after the corresponding data is uploaded. To do so, set the use-for-pruning-watermark: true in the Fullnode.yaml file as described in Archival fallback .

To enable historic data queries for the Sui Full nodes that prune old transactional data, Full node RPC implementation is configured to fallback for querying missing transactional data from a remote store.

If the information about the transaction digest, effects, events, or checkpoints is not available locally, a Full node automatically

retrieves the historical data from a cloud-based key-value store (currently managed by MystenLabs). Note that the current key-value store implementation keeps historic transactional data only: we plan to provide support for a similar setup for retrieving the historic object versions in a future release.

Sui adds new object versions to the database as part of transaction execution. This makes previous versions ready for garbage collection. However, without pruning, this can result in database performance degradation and requires large amounts of storage space. Sui identifies the objects that are eligible for pruning in each checkpoint, and then performs the pruning in the background.

You can enable pruning for a Sui node by adding the authority-store-pruning-config config to fullnode.yaml file:

Transaction pruning removes previous transactions and effects from the database. Sui periodically creates checkpoints. Each checkpoint contains the transactions that occurred during the checkpoint and their associated effects.

Sui performs transaction pruning in the background after checkpoints complete.

You can enable transaction pruning for your Full node or Validator node by adding num-epochs-to-retain-for-checkpoints to the authority-store-pruning-config config for the node:

If you prune transactions, Archival nodes can help ensure lagging peer nodes don't lose any information. For more information, see Sui Archives .

Use the examples in this section to configure your Sui Full node. You can copy the examples, and then, optionally, modify the values as appropriate for your environment.

This configuration keeps disk usage to a minimum. It is suitable for most validators. A Full node with this configuration cannot answer queries that require indexing or historic data.

This setup manages secondary indexing in addition to the latest state, but aggressively prunes historic data. A Full node with this configuration:

This configuration manages the full object history while still pruning historic transactions. A Full node with this configuration can answer all historic and indexing queries (using the transaction query fallback for transactional data), including the ones that require historic objects such as the showBalanceChanges filter of sui_getTransactionBlock() .

The main caveat is that the current setup enables transaction pruner to go ahead of object pruner . The object pruner might not be able to properly clean up the objects modified by the transactions that have been already pruned. You should closely monitor the disk space growth on a Full node with this configuration.

In addition to the regular (pruned) snapshots, Mysten Labs also maintains special RocksDB snapshots with full history of object versions available for the operators using this configuration.

Basic Sui Full node functionality

The minimal version of a Sui Full node executes all of the transactions Sui validators commit. Sui Full nodes also orchestrate the submitting of new transactions to the system:

The preceding image shows how data flows through a Full node:

A Sui Full Node stores multiple categories of data in its permanent store.

The per-epoch Sui store is beyond the scope of this topic. Sui uses the per-epoch store (resets at the start of each epoch) internally for authority and consensus operations.

Sui Full nodes store the following types of data:

Sui Full nodes support more than 40 RPC types that includes the following categories:

Refer to Access Sui Data for an overview of options to access Sui network data.

A Sui archive instance stores the full Sui transaction history since genesis in a database agnostic format. This includes information about transactions (with client authentication), effects, events, and checkpoints. As such, archival storage can be used for data auditing and for replaying historic transactions.

The current archival storage format doesn't include historic object versions.

As a Full node operator, you can enable archival fallback for your Full node by specifying the URL to upload archival data. Currently, Mysten Labs manages a Sui archive and stores it in AWS S3. To ensure a healthy network, we encourage the Sui community to set up additional archives to ensure archival data availability across the network. In a typical configuration, an archive trails behind the latest checkpoint by approximately 10 minutes.

A Full Node that starts from scratch can replay (and thus re-verify) transactions that occurred since Sui genesis from the given archive via configuring Archival Fallback in the fullnode.yaml configuration file to point to the S3 bucket that stores the archive.

A Sui Full node that fails to retrieve checkpoints from its peers via state sync protocol falls back to downloading the missing checkpoints from its pre-configured archive. This fallback enables a Full node to catch up with the rest of the system regardless of the pruning policies of its peers.

As described previously, sustainable disk usage requires Sui Full nodes to prune the information about historic object versions as well as historic transactions with the corresponding effects and events, including old checkpoint data.

Both transaction and object pruners run in the background. The logical deletion of entries from RocksDB ultimately triggers the physical compaction of data on disk, which is governed by RocksDB background jobs: the pruning effect on disk usage is not immediate and might take multiple days.

To learn more about object pruning policies, see Object pruning . You can configure the pruner in two modes:

Keeping some history (5 epochs) provides better RPC performance as it allows more lookups to happen from local disk rather than from remote storage.

To learn more about transaction pruning policies, see Transaction pruning . To configure transaction pruning, specify the num-epochs-to-retain-for-checkpoints: X config option. The checkpoints, including their transactions, effects and events are pruned up to X epochs ago. We suggest setting transaction pruning to 2 epochs.

In case your Full node is configured to upload committed information to an archive, you should ensure that pruning doesn't occur until after the corresponding data is uploaded. To do so, set the use-for-pruning-watermark: true in the Fullnode.yaml file as described in Archival fallback .

To enable historic data queries for the Sui Full nodes that prune old transactional data, Full node RPC implementation is configured to fallback for querying missing transactional data from a remote store.

If the information about the transaction digest, effects, events, or checkpoints is not available locally, a Full node automatically retrieves the historical data from a cloud-based key-value store (currently managed by MystenLabs). Note that the current key-value store implementation keeps historic transactional data only: we plan to provide support for a similar setup for retrieving the historic object versions in a future release.

Sui adds new object versions to the database as part of transaction execution. This makes previous versions ready for garbage collection. However, without pruning, this can result in database performance degradation and requires large amounts of storage space. Sui identifies the objects that are eligible for pruning in each checkpoint, and then performs the pruning in the background.

You can enable pruning for a Sui node by adding the authority-store-pruning-config config to fullnode.yaml file:

Transaction pruning removes previous transactions and effects from the database. Sui periodically creates checkpoints. Each checkpoint contains the transactions that occurred during the checkpoint and their associated effects.

Sui performs transaction pruning in the background after checkpoints complete.

You can enable transaction pruning for your Full node or Validator node by adding num-epochs-to-retain-for-checkpoints to the authority-store-pruning-config config for the node:

If you prune transactions, Archival nodes can help ensure lagging peer nodes don't lose any information. For more information, see Sui Archives .

Use the examples in this section to configure your Sui Full node. You can copy the examples, and then, optionally, modify the values as appropriate for your environment.

This configuration keeps disk usage to a minimum. It is suitable for most validators. A Full node with this configuration cannot answer queries that require indexing or historic data.

This setup manages secondary indexing in addition to the latest state, but aggressively prunes historic data. A Full node with this configuration:

This configuration manages the full object history while still pruning historic transactions. A Full node with this configuration can answer all historic and indexing queries (using the transaction query fallback for transactional data), including the ones that require historic objects such as the showBalanceChanges filter of sui_getTransactionBlock() .

The main caveat is that the current setup enables transaction pruner to go ahead of object pruner . The object pruner might not be able to properly clean up the objects modified by the transactions that have been already pruned. You should closely monitor the disk space growth on a Full node with this configuration.

In addition to the regular (pruned) snapshots, Mysten Labs also maintains special RocksDB snapshots with full history of object versions available for the operators using this configuration.

Sui Full node Data and RPC types

A Sui Full Node stores multiple categories of data in its permanent store.

The per-epoch Sui store is beyond the scope of this topic. Sui uses the per-epoch store (resets at the start of each epoch) internally for authority and consensus operations.

Sui Full nodes store the following types of data:

Sui Full nodes support more than 40 RPC types that includes the following categories:

Refer to Access Sui Data for an overview of options to access Sui network data.

A Sui archive instance stores the full Sui transaction history since genesis in a database agnostic format. This includes information about transactions (with client authentication), effects, events, and checkpoints. As such, archival storage can be used for data auditing and for replaying historic transactions.

The current archival storage format doesn't include historic object versions.

As a Full node operator, you can enable archival fallback for your Full node by specifying the URL to upload archival data. Currently, Mysten Labs manages a Sui archive and stores it in AWS S3. To ensure a healthy network, we encourage the Sui community to set up additional archives to ensure archival data availability across the network. In a typical configuration, an archive trails behind the latest checkpoint by approximately 10 minutes.

A Full Node that starts from scratch can replay (and thus re-verify) transactions that occurred since Sui genesis from the given archive via configuring Archival Fallback in the fullnode.yaml configuration file to point to the S3 bucket that stores the archive.

A Sui Full node that fails to retrieve checkpoints from its peers via state sync protocol falls back to downloading the missing checkpoints from its pre-configured archive. This fallback enables a Full node to catch up with the rest of the system regardless of the pruning policies of its peers.

As described previously, sustainable disk usage requires Sui Full nodes to prune the information about historic object versions as well as historic transactions with the corresponding effects and events, including old checkpoint data.

Both transaction and object pruners run in the background. The logical deletion of entries from RocksDB ultimately triggers the physical compaction of data on disk, which is governed by RocksDB background jobs: the pruning effect on disk usage is not immediate and might take multiple days.

To learn more about object pruning policies, see Object pruning . You can configure the pruner in two modes:

Keeping some history (5 epochs) provides better RPC performance as it allows more lookups to happen from local disk rather than from remote storage.

To learn more about transaction pruning policies, see Transaction pruning . To configure transaction pruning, specify the num-epochs-to-retain-for-checkpoints: X config option. The checkpoints, including their transactions, effects and events are pruned up to X epochs ago. We suggest setting transaction pruning to 2 epochs.

In case your Full node is configured to upload committed information to an archive, you should ensure that pruning doesn't occur until after the corresponding data is uploaded. To do so, set the use-for-pruning-watermark: true in the Fullnode.yaml file as described in Archival fallback .

To enable historic data queries for the Sui Full nodes that prune old transactional data, Full node RPC implementation is configured

to fallback for querying missing transactional data from a remote store.

If the information about the transaction digest, effects, events, or checkpoints is not available locally, a Full node automatically retrieves the historical data from a cloud-based key-value store (currently managed by MystenLabs). Note that the current key-value store implementation keeps historic transactional data only: we plan to provide support for a similar setup for retrieving the historic object versions in a future release.

Sui adds new object versions to the database as part of transaction execution. This makes previous versions ready for garbage collection. However, without pruning, this can result in database performance degradation and requires large amounts of storage space. Sui identifies the objects that are eligible for pruning in each checkpoint, and then performs the pruning in the background.

You can enable pruning for a Sui node by adding the authority-store-pruning-config config to fullnode.yaml file:

Transaction pruning removes previous transactions and effects from the database. Sui periodically creates checkpoints. Each checkpoint contains the transactions that occurred during the checkpoint and their associated effects.

Sui performs transaction pruning in the background after checkpoints complete.

You can enable transaction pruning for your Full node or Validator node by adding num-epochs-to-retain-for-checkpoints to the authority-store-pruning-config config for the node:

If you prune transactions, Archival nodes can help ensure lagging peer nodes don't lose any information. For more information, see Sui Archives .

Use the examples in this section to configure your Sui Full node. You can copy the examples, and then, optionally, modify the values as appropriate for your environment.

This configuration keeps disk usage to a minimum. It is suitable for most validators. A Full node with this configuration cannot answer queries that require indexing or historic data.

This setup manages secondary indexing in addition to the latest state, but aggressively prunes historic data. A Full node with this configuration:

This configuration manages the full object history while still pruning historic transactions. A Full node with this configuration can answer all historic and indexing queries (using the transaction query fallback for transactional data), including the ones that require historic objects such as the showBalanceChanges filter of sui_getTransactionBlock() .

The main caveat is that the current setup enables transaction pruner to go ahead of object pruner . The object pruner might not be able to properly clean up the objects modified by the transactions that have been already pruned. You should closely monitor the disk space growth on a Full node with this configuration.

In addition to the regular (pruned) snapshots, Mysten Labs also maintains special RocksDB snapshots with full history of object versions available for the operators using this configuration.

Sui Archival data

A Sui archive instance stores the full Sui transaction history since genesis in a database agnostic format. This includes information about transactions (with client authentication), effects, events, and checkpoints. As such, archival storage can be used for data auditing and for replaying historic transactions.

The current archival storage format doesn't include historic object versions.

As a Full node operator, you can enable archival fallback for your Full node by specifying the URL to upload archival data. Currently, Mysten Labs manages a Sui archive and stores it in AWS S3. To ensure a healthy network, we encourage the Sui community to set up additional archives to ensure archival data availability across the network. In a typical configuration, an archive trails behind the latest checkpoint by approximately 10 minutes.

A Full Node that starts from scratch can replay (and thus re-verify) transactions that occurred since Sui genesis from the given archive via configuring Archival Fallback in the fullnode.yaml configuration file to point to the S3 bucket that stores the archive.

A Sui Full node that fails to retrieve checkpoints from its peers via state sync protocol falls back to downloading the missing checkpoints from its pre-configured archive. This fallback enables a Full node to catch up with the rest of the system regardless of the pruning policies of its peers.

As described previously, sustainable disk usage requires Sui Full nodes to prune the information about historic object versions as well as historic transactions with the corresponding effects and events, including old checkpoint data.

Both transaction and object pruners run in the background. The logical deletion of entries from RocksDB ultimately triggers the physical compaction of data on disk, which is governed by RocksDB background jobs: the pruning effect on disk usage is not immediate and might take multiple days.

To learn more about object pruning policies, see Object pruning . You can configure the pruner in two modes:

Keeping some history (5 epochs) provides better RPC performance as it allows more lookups to happen from local disk rather than from remote storage.

To learn more about transaction pruning policies, see Transaction pruning . To configure transaction pruning, specify the num-epochs-to-retain-for-checkpoints: X config option. The checkpoints, including their transactions, effects and events are pruned up to X epochs ago. We suggest setting transaction pruning to 2 epochs.

In case your Full node is configured to upload committed information to an archive, you should ensure that pruning doesn't occur until after the corresponding data is uploaded. To do so, set the use-for-pruning-watermark: true in the Fullnode.yaml file as described in Archival fallback .

To enable historic data queries for the Sui Full nodes that prune old transactional data, Full node RPC implementation is configured to fallback for querying missing transactional data from a remote store.

If the information about the transaction digest, effects, events, or checkpoints is not available locally, a Full node automatically retrieves the historical data from a cloud-based key-value store (currently managed by MystenLabs). Note that the current key-value store implementation keeps historic transactional data only: we plan to provide support for a similar setup for retrieving the historic object versions in a future release.

Sui adds new object versions to the database as part of transaction execution. This makes previous versions ready for garbage collection. However, without pruning, this can result in database performance degradation and requires large amounts of storage space. Sui identifies the objects that are eligible for pruning in each checkpoint, and then performs the pruning in the background.

You can enable pruning for a Sui node by adding the authority-store-pruning-config config to fullnode.yaml file:

Transaction pruning removes previous transactions and effects from the database. Sui periodically creates checkpoints. Each checkpoint contains the transactions that occurred during the checkpoint and their associated effects.

Sui performs transaction pruning in the background after checkpoints complete.

You can enable transaction pruning for your Full node or Validator node by adding num-epochs-to-retain-for-checkpoints to the authority-store-pruning-config config for the node:

If you prune transactions, Archival nodes can help ensure lagging peer nodes don't lose any information. For more information, see Sui Archives .

Use the examples in this section to configure your Sui Full node. You can copy the examples, and then, optionally, modify the values as appropriate for your environment.

This configuration keeps disk usage to a minimum. It is suitable for most validators. A Full node with this configuration cannot answer queries that require indexing or historic data.

This setup manages secondary indexing in addition to the latest state, but aggressively prunes historic data. A Full node with this configuration:

This configuration manages the full object history while still pruning historic transactions. A Full node with this configuration can answer all historic and indexing queries (using the transaction query fallback for transactional data), including the ones that require historic objects such as the showBalanceChanges filter of sui_getTransactionBlock() .

The main caveat is that the current setup enables transaction pruner to go ahead of object pruner . The object pruner might not be able to properly clean up the objects modified by the transactions that have been already pruned. You should closely monitor the disk space growth on a Full node with this configuration.

In addition to the regular (pruned) snapshots, Mysten Labs also maintains special RocksDB snapshots with full history of object versions available for the operators using this configuration.

Sui Full node pruning policies

As described previously, sustainable disk usage requires Sui Full nodes to prune the information about historic object versions as well as historic transactions with the corresponding effects and events, including old checkpoint data.

Both transaction and object pruners run in the background. The logical deletion of entries from RocksDB ultimately triggers the physical compaction of data on disk, which is governed by RocksDB background jobs: the pruning effect on disk usage is not immediate and might take multiple days.

To learn more about object pruning policies, see Object pruning . You can configure the pruner in two modes:

Keeping some history (5 epochs) provides better RPC performance as it allows more lookups to happen from local disk rather than from remote storage.

To learn more about transaction pruning policies, see Transaction pruning . To configure transaction pruning, specify the num-epochs-to-retain-for-checkpoints: X config option. The checkpoints, including their transactions, effects and events are pruned up to X epochs ago. We suggest setting transaction pruning to 2 epochs.

In case your Full node is configured to upload committed information to an archive, you should ensure that pruning doesn't occur until after the corresponding data is uploaded. To do so, set the use-for-pruning-watermark: true in the Fullnode.yaml file as described in Archival fallback .

To enable historic data queries for the Sui Full nodes that prune old transactional data, Full node RPC implementation is configured to fallback for querying missing transactional data from a remote store.

If the information about the transaction digest, effects, events, or checkpoints is not available locally, a Full node automatically retrieves the historical data from a cloud-based key-value store (currently managed by MystenLabs). Note that the current key-value store implementation keeps historic transactional data only: we plan to provide support for a similar setup for retrieving the historic object versions in a future release.

Sui adds new object versions to the database as part of transaction execution. This makes previous versions ready for garbage collection. However, without pruning, this can result in database performance degradation and requires large amounts of storage space. Sui identifies the objects that are eligible for pruning in each checkpoint, and then performs the pruning in the background.

You can enable pruning for a Sui node by adding the authority-store-pruning-config config to fullnode.yaml file:

Transaction pruning removes previous transactions and effects from the database. Sui periodically creates checkpoints. Each checkpoint contains the transactions that occurred during the checkpoint and their associated effects.

Sui performs transaction pruning in the background after checkpoints complete.

You can enable transaction pruning for your Full node or Validator node by adding num-epochs-to-retain-for-checkpoints to the authority-store-pruning-config config for the node:

If you prune transactions, Archival nodes can help ensure lagging peer nodes don't lose any information. For more information, see Sui Archives .

Use the examples in this section to configure your Sui Full node. You can copy the examples, and then, optionally, modify the values as appropriate for your environment.

This configuration keeps disk usage to a minimum. It is suitable for most validators. A Full node with this configuration cannot answer queries that require indexing or historic data.

This setup manages secondary indexing in addition to the latest state, but aggressively prunes historic data. A Full node with this configuration:

This configuration manages the full object history while still pruning historic transactions. A Full node with this configuration can answer all historic and indexing queries (using the transaction query fallback for transactional data), including the ones that require historic objects such as the showBalanceChanges filter of sui_getTransactionBlock() .

The main caveat is that the current setup enables transaction pruner to go ahead of object pruner . The object pruner might not be able to properly clean up the objects modified by the transactions that have been already pruned. You should closely monitor the disk space growth on a Full node with this configuration.

In addition to the regular (pruned) snapshots, Mysten Labs also maintains special RocksDB snapshots with full history of object versions available for the operators using this configuration.

Sui Full node key-value store backup

To enable historic data queries for the Sui Full nodes that prune old transactional data, Full node RPC implementation is configured to fallback for querying missing transactional data from a remote store.

If the information about the transaction digest, effects, events, or checkpoints is not available locally, a Full node automatically retrieves the historical data from a cloud-based key-value store (currently managed by MystenLabs). Note that the current key-value store implementation keeps historic transactional data only: we plan to provide support for a similar setup for retrieving the historic object versions in a future release.

Sui adds new object versions to the database as part of transaction execution. This makes previous versions ready for garbage collection. However, without pruning, this can result in database performance degradation and requires large amounts of storage space. Sui identifies the objects that are eligible for pruning in each checkpoint, and then performs the pruning in the background.

You can enable pruning for a Sui node by adding the authority-store-pruning-config config to fullnode.yaml file:

Transaction pruning removes previous transactions and effects from the database. Sui periodically creates checkpoints. Each checkpoint contains the transactions that occurred during the checkpoint and their associated effects.

Sui performs transaction pruning in the background after checkpoints complete.

You can enable transaction pruning for your Full node or Validator node by adding num-epochs-to-retain-for-checkpoints to the authority-store-pruning-config config for the node:

If you prune transactions, Archival nodes can help ensure lagging peer nodes don't lose any information. For more information, see Sui Archives .

Use the examples in this section to configure your Sui Full node. You can copy the examples, and then, optionally, modify the values as appropriate for your environment.

This configuration keeps disk usage to a minimum. It is suitable for most validators. A Full node with this configuration cannot answer queries that require indexing or historic data.

This setup manages secondary indexing in addition to the latest state, but aggressively prunes historic data. A Full node with this configuration:

This configuration manages the full object history while still pruning historic transactions. A Full node with this configuration can answer all historic and indexing queries (using the transaction query fallback for transactional data), including the ones that require historic objects such as the showBalanceChanges filter of sui_getTransactionBlock() .

The main caveat is that the current setup enables transaction pruner to go ahead of object pruner . The object pruner might not be able to properly clean up the objects modified by the transactions that have been already pruned. You should closely monitor the disk space growth on a Full node with this configuration.

In addition to the regular (pruned) snapshots, Mysten Labs also maintains special RocksDB snapshots with full history of object versions available for the operators using this configuration.

Object pruning

Sui adds new object versions to the database as part of transaction execution. This makes previous versions ready for garbage collection. However, without pruning, this can result in database performance degradation and requires large amounts of storage space. Sui identifies the objects that are eligible for pruning in each checkpoint, and then performs the pruning in the background.

You can enable pruning for a Sui node by adding the authority-store-pruning-config config to fullnode.yaml file:

Transaction pruning removes previous transactions and effects from the database. Sui periodically creates checkpoints. Each checkpoint contains the transactions that occurred during the checkpoint and their associated effects.

Sui performs transaction pruning in the background after checkpoints complete.

You can enable transaction pruning for your Full node or Validator node by adding num-epochs-to-retain-for-checkpoints to the

authority-store-pruning-config config for the node:

If you prune transactions, Archival nodes can help ensure lagging peer nodes don't lose any information. For more information, see Sui Archives .

Use the examples in this section to configure your Sui Full node. You can copy the examples, and then, optionally, modify the values as appropriate for your environment.

This configuration keeps disk usage to a minimum. It is suitable for most validators. A Full node with this configuration cannot answer queries that require indexing or historic data.

This setup manages secondary indexing in addition to the latest state, but aggressively prunes historic data. A Full node with this configuration:

This configuration manages the full object history while still pruning historic transactions. A Full node with this configuration can answer all historic and indexing queries (using the transaction query fallback for transactional data), including the ones that require historic objects such as the showBalanceChanges filter of sui_getTransactionBlock() .

The main caveat is that the current setup enables transaction pruner to go ahead of object pruner . The object pruner might not be able to properly clean up the objects modified by the transactions that have been already pruned. You should closely monitor the disk space growth on a Full node with this configuration.

In addition to the regular (pruned) snapshots, Mysten Labs also maintains special RocksDB snapshots with full history of object versions available for the operators using this configuration.

Transaction pruning

Transaction pruning removes previous transactions and effects from the database. Sui periodically creates checkpoints. Each checkpoint contains the transactions that occurred during the checkpoint and their associated effects.

Sui performs transaction pruning in the background after checkpoints complete.

You can enable transaction pruning for your Full node or Validator node by adding num-epochs-to-retain-for-checkpoints to the authority-store-pruning-config config for the node:

If you prune transactions, Archival nodes can help ensure lagging peer nodes don't lose any information. For more information, see Sui Archives .

Use the examples in this section to configure your Sui Full node. You can copy the examples, and then, optionally, modify the values as appropriate for your environment.

This configuration keeps disk usage to a minimum. It is suitable for most validators. A Full node with this configuration cannot answer queries that require indexing or historic data.

This setup manages secondary indexing in addition to the latest state, but aggressively prunes historic data. A Full node with this configuration:

This configuration manages the full object history while still pruning historic transactions. A Full node with this configuration can answer all historic and indexing queries (using the transaction query fallback for transactional data), including the ones that require historic objects such as the showBalanceChanges filter of sui_getTransactionBlock() .

The main caveat is that the current setup enables transaction pruner to go ahead of object pruner . The object pruner might not be able to properly clean up the objects modified by the transactions that have been already pruned. You should closely monitor the disk space growth on a Full node with this configuration.

In addition to the regular (pruned) snapshots, Mysten Labs also maintains special RocksDB snapshots with full history of object versions available for the operators using this configuration.

Pruning policy examples

Use the examples in this section to configure your Sui Full node. You can copy the examples, and then, optionally, modify the values as appropriate for your environment.

This configuration keeps disk usage to a minimum. It is suitable for most validators. A Full node with this configuration cannot

answer queries that require indexing or historic data.

This setup manages secondary indexing in addition to the latest state, but aggressively prunes historic data. A Full node with this configuration:

This configuration manages the full object history while still pruning historic transactions. A Full node with this configuration can answer all historic and indexing queries (using the transaction query fallback for transactional data), including the ones that require historic objects such as the showBalanceChanges filter of sui_getTransactionBlock() .

The main caveat is that the current setup enables transaction pruner to go ahead of object pruner . The object pruner might not be able to properly clean up the objects modified by the transactions that have been already pruned. You should closely monitor the disk space growth on a Full node with this configuration.

In addition to the regular (pruned) snapshots, Mysten Labs also maintains special RocksDB snapshots with full history of object versions available for the operators using this configuration.