Sui Bridge Validator Node Configuration

Running a Bridge Validator Node (Bridge Node) requires registering your node with the bridge committee. Correct configuration of your node ensures optimal performance and valid metrics data. Follow this topic to make sure your Bridge Node is set up properly.

To set up and run a Bridge Node, you need to install sui and sui-bridge-cli . You can install them using one of the following options:

Install from tip of main :

Install with a commit sha:

To join the network you must first register with the bridge validator committee.

The required metadata includes two things:

To create a BridgeAuthorityKey , run

where is the location to write the key pair to.

It's highly recommended you create a new key pair in a secure environment (for example, in the same machine where your node runs) to avoid key compromise.

After you have both authority key file and REST API URL ready, you can register them by using Sui CLI:

If you keep your validator account key in cold storage or you want to perform offline signing, use flags --print-only and --validator-address (with the value for the validator address). This prints serialized unsigned transaction bytes, then you can use your preferred signing process to produce signed bytes.

Run the following command to execute it:

Both key and URL are changeable before the committee is finalized . If you wish to update metadata, simply rerun sui validator register-bridge-committee .

To double check you registered the correct metadata on chain, run

Use the following command to update bridge node URL:

Refer to offline signing section in this page for how to sign the transaction offline.

Authoritiy key rotation is not supported yet.

You have several options when configuring your Bridge Node for performance and metrics monitoring. Follow the instructions that follow to configure your node for best results in your environment.

Suggested hardware requirements:

To protect against distributed denial of service (DDoS) attacks and similar attacks intended to expend validator resources, you must provide rate limit protection for the bridge server.

In addition to protection, this gives node operators fine-grained control over the rate of requests they receive, and observability into those requests.

The currently recommended rate limit is 50 requests/second per unique IP .

You can use a managed cloud service, for example:

It's also possible to use an open source load balancer, such as HAProxy for a practical, IP-based rate limit.

A shortened example HAProxy configuration looks like the following:

If choosing to use an open source load balancing option, make sure to set up metrics collection and alerting on the service.

Use sui-bridge-cli command to create a template. If you want to run BridgeClient (see the following section), pass --run-client as a parameter.

The generated configuration includes the following parameters:

With run-client: true , you can find these additional fields in the generated config:

BridgeClient orchestrates bridge transfer requests. It is optional to run for a BridgeNode . BridgeClient submits transaction on the Sui network. Thus when it's enabled, you need a Sui account key with enough SUI balance.

To enable bridge_client feature on a BridgeNode , set the following parameters in BridgeNodeConfig :

To create a BridgeClient key pair, run

This prints the newly created Sui Address. Then we need to fund this address with some SUI for operations.

Build or install Bridge Node in one of the following ways:

Running Bridge Node is similar to running a Sui node using systemd or Ansible. The command to start the Bridge Node is:

Bridge Node listens for TCP connections over port 9191 (or the preferred port in the configuration file). You must allow incoming connections for that port on the host that is running Bridge Node.

Test ingress with curl on a remote machine and expect a 200 response:

Use uptime to check if the node is running.

You can find a full list of Bridge Node metrics and their descriptions in the sui-bridge crate .

In this case Bridge Node runs as a passive observer and does not proactively poll on-chain activities. Important metrics to monitor in this case are the request handling metrics, such as:

In this case, Bridge Client is toggled on and syncs with blockchains proactively. The best metrics to track progress are:

bridge_gas_coin_balance is also a critical metric to track the balance of your client gas coin, and top up after it dips below a certain threshold.

The Bridge Nodes can push metrics to the remote proxy for network-level observability.

To enable metrics push, set the following parameters in BridgeNodeConfig :

The proxy authenticates pushed metrics by using the metrics key pair. It is similar to sui-node pushing metrics with NetworkKey . Unlike NetworkKey , the Bridge Node metrics key is not recorded on chain and can be ephemeral. The metrics key is loaded from the metrics-key-pair field in BridgeNodeConfig if provided, otherwise a new key pair is generated on the fly. The proxy queries node public keys periodically by hitting the metrics public API key of each node.

When Bridge Node starts, it might log this line once:

This is okay to ignore as long as it does not persist. Otherwise, try:

and make sure the public key is correctly returned.

Prerequisites

To set up and run a Bridge Node, you need to install sui and sui-bridge-cli . You can install them using one of the following options:

Install from tip of main :

Install with a commit sha:

To join the network you must first register with the bridge validator committee.

The required metadata includes two things:

To create a BridgeAuthorityKey , run

where is the location to write the key pair to.

It's highly recommended you create a new key pair in a secure environment (for example, in the same machine where your node

runs) to avoid key compromise.

After you have both authority key file and REST API URL ready, you can register them by using Sui CLI:

If you keep your validator account key in cold storage or you want to perform offline signing, use flags --print-only and --validator-address (with the value for the validator address). This prints serialized unsigned transaction bytes, then you can use your preferred signing process to produce signed bytes.

Run the following command to execute it:

Both key and URL are changeable before the committee is finalized . If you wish to update metadata, simply rerun sui validator register-bridge-committee .

To double check you registered the correct metadata on chain, run

Use the following command to update bridge node URL:

Refer to offline signing section in this page for how to sign the transaction offline.

Authoritiy key rotation is not supported yet.

You have several options when configuring your Bridge Node for performance and metrics monitoring. Follow the instructions that follow to configure your node for best results in your environment.

Suggested hardware requirements:

To protect against distributed denial of service (DDoS) attacks and similar attacks intended to expend validator resources, you must provide rate limit protection for the bridge server.

In addition to protection, this gives node operators fine-grained control over the rate of requests they receive, and observability into those requests.

The currently recommended rate limit is 50 requests/second per unique IP .

You can use a managed cloud service, for example:

It's also possible to use an open source load balancer, such as HAProxy for a practical, IP-based rate limit.

A shortened example HAProxy configuration looks like the following:

If choosing to use an open source load balancing option, make sure to set up metrics collection and alerting on the service.

Use sui-bridge-cli command to create a template. If you want to run BridgeClient (see the following section), pass --run-client as a parameter.

The generated configuration includes the following parameters:

With run-client: true , you can find these additional fields in the generated config:

BridgeClient orchestrates bridge transfer requests. It is optional to run for a BridgeNode . BridgeClient submits transaction on the Sui network. Thus when it's enabled, you need a Sui account key with enough SUI balance.

To enable bridge_client feature on a BridgeNode , set the following parameters in BridgeNodeConfig :

To create a BridgeClient key pair, run

This prints the newly created Sui Address. Then we need to fund this address with some SUI for operations.

Build or install Bridge Node in one of the following ways:

Running Bridge Node is similar to running a Sui node using systemd or Ansible. The command to start the Bridge Node is:

Bridge Node listens for TCP connections over port 9191 (or the preferred port in the configuration file). You must allow incoming connections for that port on the host that is running Bridge Node.

Test ingress with curl on a remote machine and expect a 200 response:

Use uptime to check if the node is running.

You can find a full list of Bridge Node metrics and their descriptions in the sui-bridge crate .

In this case Bridge Node runs as a passive observer and does not proactively poll on-chain activities. Important metrics to monitor in this case are the request handling metrics, such as:

In this case, Bridge Client is toggled on and syncs with blockchains proactively. The best metrics to track progress are:

bridge_gas_coin_balance is also a critical metric to track the balance of your client gas coin, and top up after it dips below a certain threshold.

The Bridge Nodes can push metrics to the remote proxy for network-level observability.

To enable metrics push, set the following parameters in BridgeNodeConfig :

The proxy authenticates pushed metrics by using the metrics key pair. It is similar to sui-node pushing metrics with NetworkKey . Unlike NetworkKey , the Bridge Node metrics key is not recorded on chain and can be ephemeral. The metrics key is loaded from the metrics-key-pair field in BridgeNodeConfig if provided, otherwise a new key pair is generated on the fly. The proxy queries node public keys periodically by hitting the metrics public API key of each node.

When Bridge Node starts, it might log this line once:

This is okay to ignore as long as it does not persist. Otherwise, try:

and make sure the public key is correctly returned.

Committee registration

To join the network you must first register with the bridge validator committee.

The required metadata includes two things:

To create a BridgeAuthorityKey , run

where is the location to write the key pair to.

It's highly recommended you create a new key pair in a secure environment (for example, in the same machine where your node runs) to avoid key compromise.

After you have both authority key file and REST API URL ready, you can register them by using Sui CLI:

If you keep your validator account key in cold storage or you want to perform offline signing, use flags --print-only and --validator-address (with the value for the validator address). This prints serialized unsigned transaction bytes, then you can use your preferred signing process to produce signed bytes.

Run the following command to execute it:

Both key and URL are changeable before the committee is finalized . If you wish to update metadata, simply rerun sui validator register-bridge-committee .

To double check you registered the correct metadata on chain, run

Use the following command to update bridge node URL:

Refer to offline signing section in this page for how to sign the transaction offline.

Authoritiy key rotation is not supported yet.

You have several options when configuring your Bridge Node for performance and metrics monitoring. Follow the instructions that follow to configure your node for best results in your environment.

Suggested hardware requirements:

To protect against distributed denial of service (DDoS) attacks and similar attacks intended to expend validator resources, you must

provide rate limit protection for the bridge server.

In addition to protection, this gives node operators fine-grained control over the rate of requests they receive, and observability into those requests.

The currently recommended rate limit is 50 requests/second per unique IP .

You can use a managed cloud service, for example:

It's also possible to use an open source load balancer, such as HAProxy for a practical, IP-based rate limit.

A shortened example HAProxy configuration looks like the following:

If choosing to use an open source load balancing option, make sure to set up metrics collection and alerting on the service.

Use sui-bridge-cli command to create a template. If you want to run BridgeClient (see the following section), pass --run-client as a parameter.

The generated configuration includes the following parameters:

With run-client: true , you can find these additional fields in the generated config:

BridgeClient orchestrates bridge transfer requests. It is optional to run for a BridgeNode . BridgeClient submits transaction on the Sui network. Thus when it's enabled, you need a Sui account key with enough SUI balance.

To enable bridge_client feature on a BridgeNode , set the following parameters in BridgeNodeConfig :

To create a BridgeClient key pair, run

This prints the newly created Sui Address. Then we need to fund this address with some SUI for operations.

Build or install Bridge Node in one of the following ways:

Running Bridge Node is similar to running a Sui node using systemd or Ansible. The command to start the Bridge Node is:

Bridge Node listens for TCP connections over port 9191 (or the preferred port in the configuration file). You must allow incoming connections for that port on the host that is running Bridge Node.

Test ingress with curl on a remote machine and expect a 200 response:

Use uptime to check if the node is running.

You can find a full list of Bridge Node metrics and their descriptions in the sui-bridge crate .

In this case Bridge Node runs as a passive observer and does not proactively poll on-chain activities. Important metrics to monitor in this case are the request handling metrics, such as:

In this case, Bridge Client is toggled on and syncs with blockchains proactively. The best metrics to track progress are:

bridge_gas_coin_balance is also a critical metric to track the balance of your client gas coin, and top up after it dips below a certain threshold.

The Bridge Nodes can push metrics to the remote proxy for network-level observability.

To enable metrics push, set the following parameters in BridgeNodeConfig :

The proxy authenticates pushed metrics by using the metrics key pair. It is similar to sui-node pushing metrics with NetworkKey . Unlike NetworkKey , the Bridge Node metrics key is not recorded on chain and can be ephemeral. The metrics key is loaded from the metrics-key-pair field in BridgeNodeConfig if provided, otherwise a new key pair is generated on the fly. The proxy queries node public keys periodically by hitting the metrics public API key of each node.

When Bridge Node starts, it might log this line once:

This is okay to ignore as long as it does not persist. Otherwise, try:

and make sure the public key is correctly returned.

Update metadata (after committee is finalized)

Use the following command to update bridge node URL:

Refer to offline signing section in this page for how to sign the transaction offline.

Authoritiy key rotation is not supported yet.

You have several options when configuring your Bridge Node for performance and metrics monitoring. Follow the instructions that follow to configure your node for best results in your environment.

Suggested hardware requirements:

To protect against distributed denial of service (DDoS) attacks and similar attacks intended to expend validator resources, you must provide rate limit protection for the bridge server.

In addition to protection, this gives node operators fine-grained control over the rate of requests they receive, and observability into those requests.

The currently recommended rate limit is 50 requests/second per unique IP .

You can use a managed cloud service, for example:

It's also possible to use an open source load balancer, such as HAProxy for a practical, IP-based rate limit.

A shortened example HAProxy configuration looks like the following:

If choosing to use an open source load balancing option, make sure to set up metrics collection and alerting on the service.

Use sui-bridge-cli command to create a template. If you want to run BridgeClient (see the following section), pass --run-client as a parameter.

The generated configuration includes the following parameters:

With run-client: true , you can find these additional fields in the generated config:

BridgeClient orchestrates bridge transfer requests. It is optional to run for a BridgeNode . BridgeClient submits transaction on the Sui network. Thus when it's enabled, you need a Sui account key with enough SUI balance.

To enable bridge_client feature on a BridgeNode , set the following parameters in BridgeNodeConfig :

To create a BridgeClient key pair, run

This prints the newly created Sui Address. Then we need to fund this address with some SUI for operations.

Build or install Bridge Node in one of the following ways:

Running Bridge Node is similar to running a Sui node using systemd or Ansible. The command to start the Bridge Node is:

Bridge Node listens for TCP connections over port 9191 (or the preferred port in the configuration file). You must allow incoming connections for that port on the host that is running Bridge Node.

Test ingress with curl on a remote machine and expect a 200 response:

Use uptime to check if the node is running.

You can find a full list of Bridge Node metrics and their descriptions in the sui-bridge crate .

In this case Bridge Node runs as a passive observer and does not proactively poll on-chain activities. Important metrics to monitor in this case are the request handling metrics, such as:

In this case, Bridge Client is toggled on and syncs with blockchains proactively. The best metrics to track progress are:

bridge_gas_coin_balance is also a critical metric to track the balance of your client gas coin, and top up after it dips below a certain threshold.

The Bridge Nodes can push metrics to the remote proxy for network-level observability.

To enable metrics push, set the following parameters in BridgeNodeConfig :

The proxy authenticates pushed metrics by using the metrics key pair. It is similar to sui-node pushing metrics with NetworkKey . Unlike NetworkKey , the Bridge Node metrics key is not recorded on chain and can be ephemeral. The metrics key is loaded from the metrics-key-pair field in BridgeNodeConfig if provided, otherwise a new key pair is generated on the fly. The proxy queries node public keys periodically by hitting the metrics public API key of each node.

When Bridge Node starts, it might log this line once:

This is okay to ignore as long as it does not persist. Otherwise, try:

and make sure the public key is correctly returned.

Bridge Node

You have several options when configuring your Bridge Node for performance and metrics monitoring. Follow the instructions that follow to configure your node for best results in your environment.

Suggested hardware requirements:

To protect against distributed denial of service (DDoS) attacks and similar attacks intended to expend validator resources, you must provide rate limit protection for the bridge server.

In addition to protection, this gives node operators fine-grained control over the rate of requests they receive, and observability into those requests.

The currently recommended rate limit is 50 requests/second per unique IP .

You can use a managed cloud service, for example:

It's also possible to use an open source load balancer, such as HAProxy for a practical, IP-based rate limit.

A shortened example HAProxy configuration looks like the following:

If choosing to use an open source load balancing option, make sure to set up metrics collection and alerting on the service.

Use sui-bridge-cli command to create a template. If you want to run BridgeClient (see the following section), pass --run-client as a parameter.

The generated configuration includes the following parameters:

With run-client: true , you can find these additional fields in the generated config:

BridgeClient orchestrates bridge transfer requests. It is optional to run for a BridgeNode . BridgeClient submits transaction on the Sui network. Thus when it's enabled, you need a Sui account key with enough SUI balance.

To enable bridge_client feature on a BridgeNode , set the following parameters in BridgeNodeConfig :

To create a BridgeClient key pair, run

This prints the newly created Sui Address. Then we need to fund this address with some SUI for operations.

Build or install Bridge Node in one of the following ways:

Running Bridge Node is similar to running a Sui node using systemd or Ansible. The command to start the Bridge Node is:

Bridge Node listens for TCP connections over port 9191 (or the preferred port in the configuration file). You must allow incoming connections for that port on the host that is running Bridge Node.

Test ingress with curl on a remote machine and expect a 200 response:

Use uptime to check if the node is running.

You can find a full list of Bridge Node metrics and their descriptions in the sui-bridge crate .

In this case Bridge Node runs as a passive observer and does not proactively poll on-chain activities. Important metrics to monitor in this case are the request handling metrics, such as:

In this case, Bridge Client is toggled on and syncs with blockchains proactively. The best metrics to track progress are:

bridge_gas_coin_balance is also a critical metric to track the balance of your client gas coin, and top up after it dips below a certain threshold.

The Bridge Nodes can push metrics to the remote proxy for network-level observability.

To enable metrics push, set the following parameters in BridgeNodeConfig :

The proxy authenticates pushed metrics by using the metrics key pair. It is similar to sui-node pushing metrics with NetworkKey . Unlike NetworkKey , the Bridge Node metrics key is not recorded on chain and can be ephemeral. The metrics key is loaded from the metrics-key-pair field in BridgeNodeConfig if provided, otherwise a new key pair is generated on the fly. The proxy queries node public keys periodically by hitting the metrics public API key of each node.

When Bridge Node starts, it might log this line once:

This is okay to ignore as long as it does not persist. Otherwise, try:

and make sure the public key is correctly returned.