Network Topologies

This page discusses network topologies possible with SHRINE, and how to configure SHRINE nodes to achieve them. In general, they're presented in order from simplest to most complex. This list is almost certainly not exhaustive.

Detailed information on SHRINE's config file format is here.

Single node

This configuration creates a network of one: a single SHRINE node, backed by a single instance of i2b2, that accepts queries and exposes a web client. This is the configuration produced by SHRINE's install scripts when run against a stock i2b2 VM.

shrine.conf:

```
shrine {
  ...

  queryEntryPoint {
    ...
  }

  hub {
    ...

    shouldQuerySelf = true
  }

  adapter {
    ...
  }
}
```

The shrine.queryEntryPoint block must be present to allow initiating queries from this node.

shrine.hub.shouldQuerySelf tells SHRINE to query the local adapter when broadcasting queries. In this case, the only place queries will be broadcast to is the current node.

The shrine.adapter block must be present for SHRINE to act as an adapter.

Detailed information on SHRINE's config file format is here.

Firewall considerations

A single-node network doesn't require any firewall openings (beyond what's necessary to expose a web client), because there are no inter-node communications.

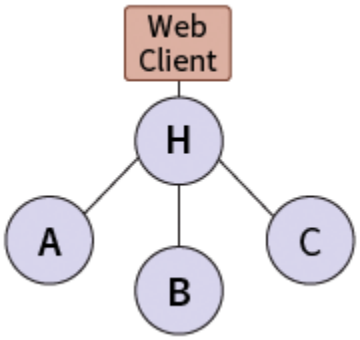
Cert considerations

A single-node network doesn't require any cert exchanges, because there are no inter-node communications.

For SHRINE ACT Network

If you are part of the ACT network, do not attempt to set up a single, standalone node. Configure your SHRINE host to join the ACT hub of your tier as a remote site.

Hub-and-spoke, single web client



This configuration creates a hub-and-spoke network with a single hub node H, and spoke nodes A, B, and C. A single web client is exposed at the hub; queries originate and terminate at the hub. The spoke nodes only respond to queries.

Hub's shrine.conf

```
shrine {
  ...

  queryEntryPoint {
    ...
  }

  hub {
    ...

    downstreamNodes {
      "Node A" = "http://nodeA.example.com/shrine/rest/adapter/requests"
      "Descriptive name for node B" = "http://nodeB.example.com/shrine/rest/adapter/requests"
      "Human-readable name for node C (can be anything)" = "http://nodeC.example.com/shrine/rest/adapter
/requests"
    }
  }

  // NB: no adapter { } block
}
```

Here, we have a shrine.queryEntryPoint block (so that queries may originate at the hub), and a shrine.hub.downstreamNodes block, so the hub knows who to broadcast queries to. The hub does not have a shrine.adapter block, since it's not queryable.

Each spoke's shrine.conf

```
shrine {
  ...

  adapter {
    ...
  }
}
```

At each spoke, there should be no shrine.queryEntryPoint block (since queries do not originate at the spokes) and no shrine.hub block (since the spokes are not hubs). A shrine.adapter block is required, since the spokes are queryable.

Detailed information on SHRINE's config file format is here.

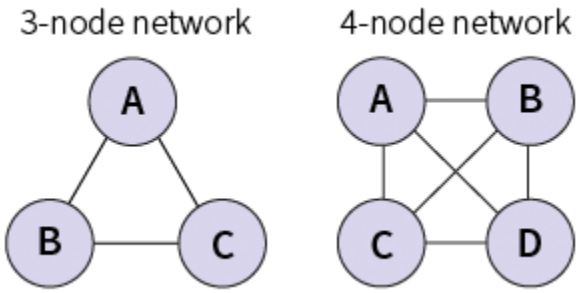
Firewall considerations

A hub-and-spoke network requires firewall openings to expose the hub's web client and to allow outbound HTTP requests from the hub to the spokes. For a network with N spokes, N unidirectional firewall openings are required.

Cert considerations

A hub-and-spoke network requires cert exchanges between the hub and each spoke. If there are N spokes, N cert exchanges are necessary.

Fully-meshed, each node can originate queries (Deprecated)



This configuration creates a fully-meshed network where each node can initate queries. It is similar to the single-node-network outlined above, but at each node, all the other nodes are listed in shrine.hub.downstreamNodes. The SHRINE team intends to stop supporting this topology in a future release. Do not create new fully meshed networks.

shrine.conf

Assuming a 3-node netowrk with nodes A, B, and C:

```
//Node A:

shrine {
  ...

  queryEntryPoint { ... }

  hub {
    ...

    shouldQuerySelf = true

    downstreamNodes {
      "node B" = "http://nodeB.example.com/shrine/rest/adapter/requests"
      "node C" = "http://nodeC.example.com/shrine/rest/adapter/requests"
    }
  }

  adapter { ... }
}

//Node B:

shrine {
  ...

  queryEntryPoint { ... }

  hub {
    ...

    shouldQuerySelf = true

    downstreamNodes {
      "node A" = "http://nodeA.example.com/shrine/rest/adapter/requests"
      "node C" = "http://nodeC.example.com/shrine/rest/adapter/requests"
    }
  }

  adapter { ... }
}

//Node C:

shrine {
  ...

  queryEntryPoint { ... }

  hub {
    ...

    shouldQuerySelf = true

    downstreamNodes {
      "node A" = "http://nodeA.example.com/shrine/rest/adapter/requests"
      "node B" = "http://nodeB.example.com/shrine/rest/adapter/requests"
    }
  }

  adapter { ... }
}
```

Each node has a shrine.queryEntryPoint block (because each node can originate queries), each node has a shrine.adapter block (because each node is queryable), and each node has a shrine.hub block (because each node broadcasts queries to the others). Another way to think about this toplogy is that an N-node fully-meshed network is comprised of N hubs, each broadcasting to the other N - 1 nodes.

Detailed information on SHRINE's config file format is here.

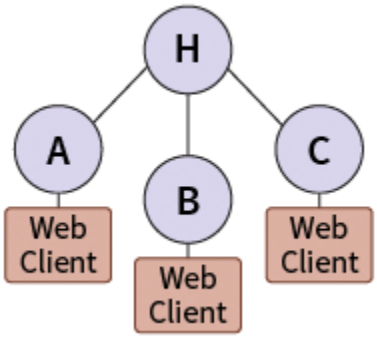
Firewall considerations

A fully-meshed network requires bidirectional firewall openings between each pair of nodes. For N nodes, the number of bidirectional firewall openings required is (N^2 - N) / 2.

Cert considerations

A fully-meshed network requires cert exchanges between each pair of nodes. For N nodes, the number of cert exchanges required is (N^2 - N) / 2.

Hub-and-spoke, web clients at each spoke



This configuration creates a hub-and-spoke network with a single hub node H, and spoke nodes A, B, and C. A web client is exposed at each spoke. Each spoke also responds to all incoming queries by acting as an adapter.

Having a single hub node reduces the number of firewall openings and cert exchanges required compared to a fully-meshed network, while allowing spoke nodes to expose their own web clients for political, branding, or other purposes.

Hub's shrine.conf

```
shrine {
  ...

  hub {
    ...

    downstreamNodes {
      "Node A" = "http://nodeA.example.com/shrine/rest/adapter/requests"
      "Node B" = "http://nodeB.example.com/shrine/rest/adapter/requests"
      "Node C" = "http://nodeC.example.com/shrine/rest/adapter/requests"
    }
  }

  // NB: no adapter { } block
}
```

Here, we have a shrine.queryEntryPoint block (so that queries may originate at the hub), and a shrine.hub.downstreamNodes block, so the hub knows who to broadcast queries to. The hub does not have a shrine.adapter block, since it's not queryable.

Each spoke's shrine.conf

```
shrine {
  ...

  queryEntryPoint {
    ...
    broadcasterServiceEndpoint {
      url = "http://hub.example.com/shrine/rest/broadcaster/broadcast"
      ...
    }
  }

  adapter {
    ...
  }
}
```

At each spoke, there should be a shrine.queryEntryPoint block that points to the hub (since each spoke originates queries, but routes them through the hub) and no shrine.hub block (since the spokes are not hubs). A shrine.adapter block is required, since the spokes are queryable.

Detailed information on SHRINE's config file format is here.

Firewall considerations

A hub-and-spoke network requires bidirectional firewall openings between the hub and all the spokes. For a network with N spokes, N bidirectional firewall openings are required.

Cert considerations

A hub-and-spoke network requires cert exchanges between the hub and each spoke. If there are N spokes, N cert exchanges are necessary.