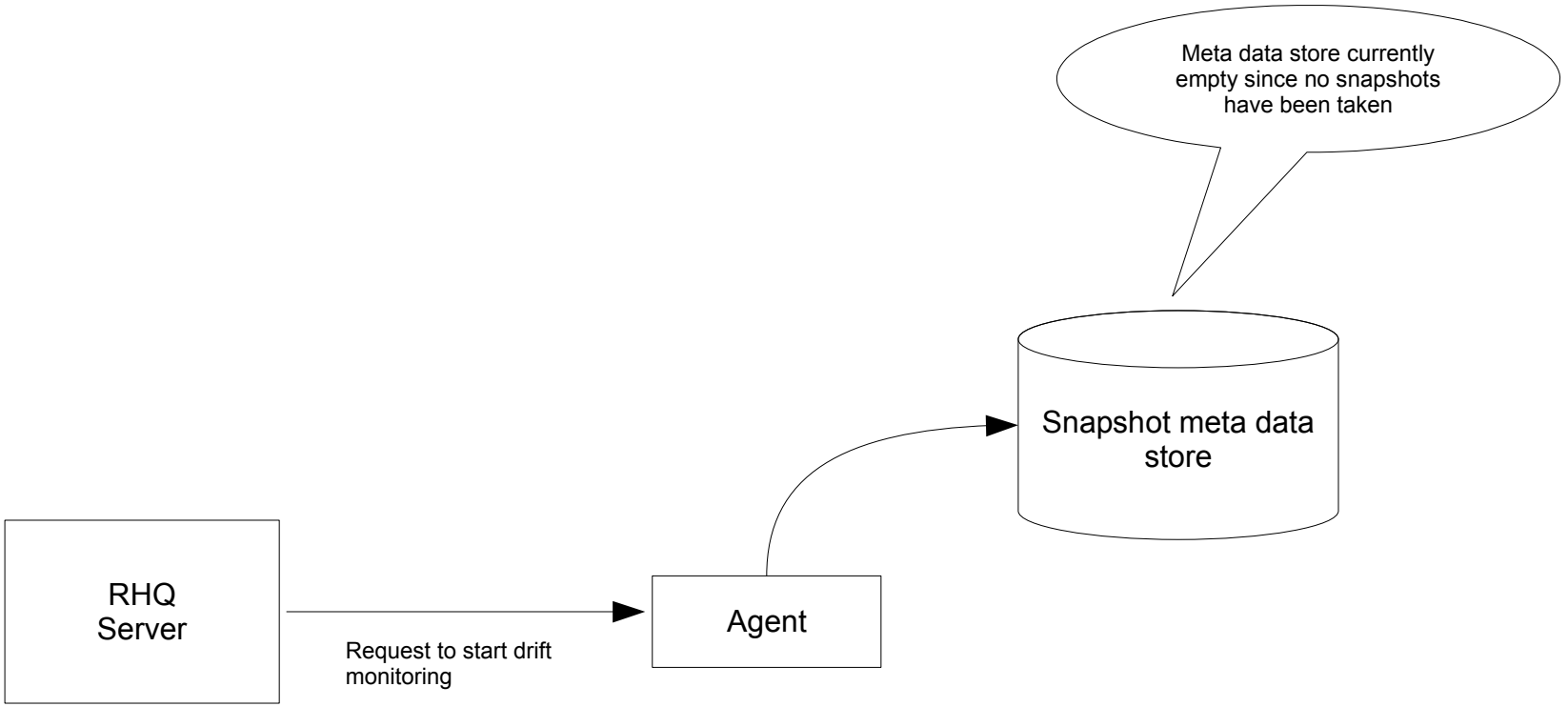
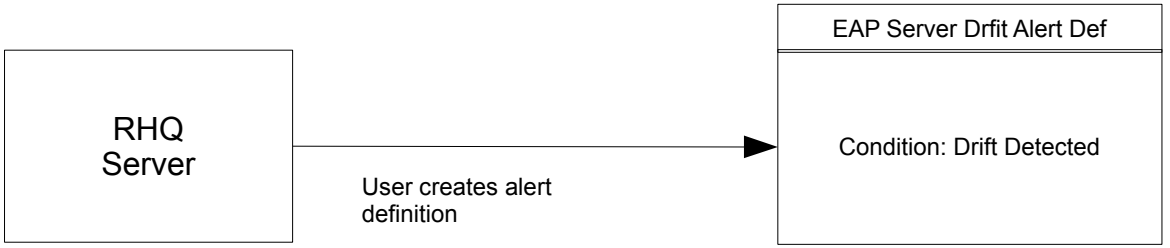
U1: Start drift monitoring for EAP server

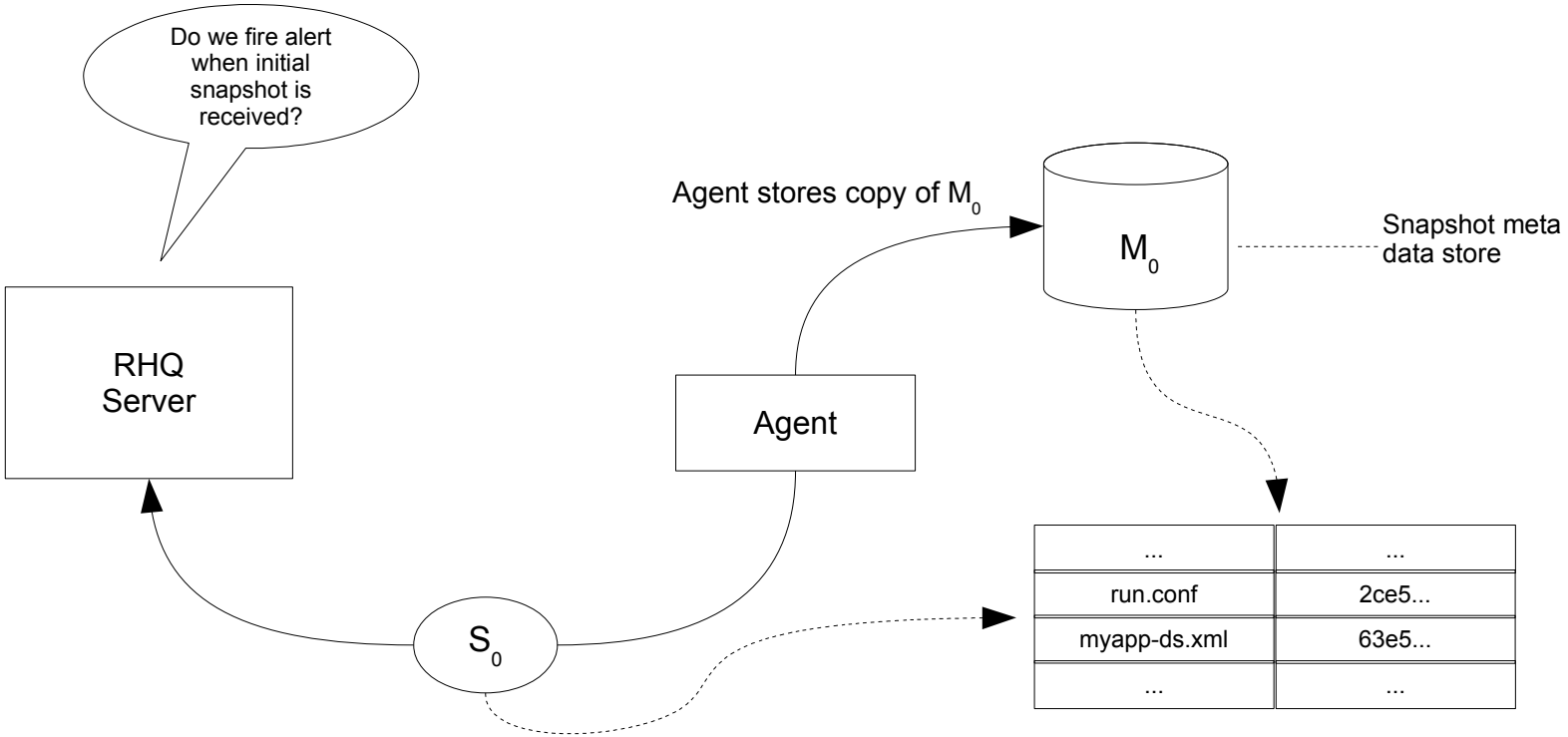


Create drift alert definition



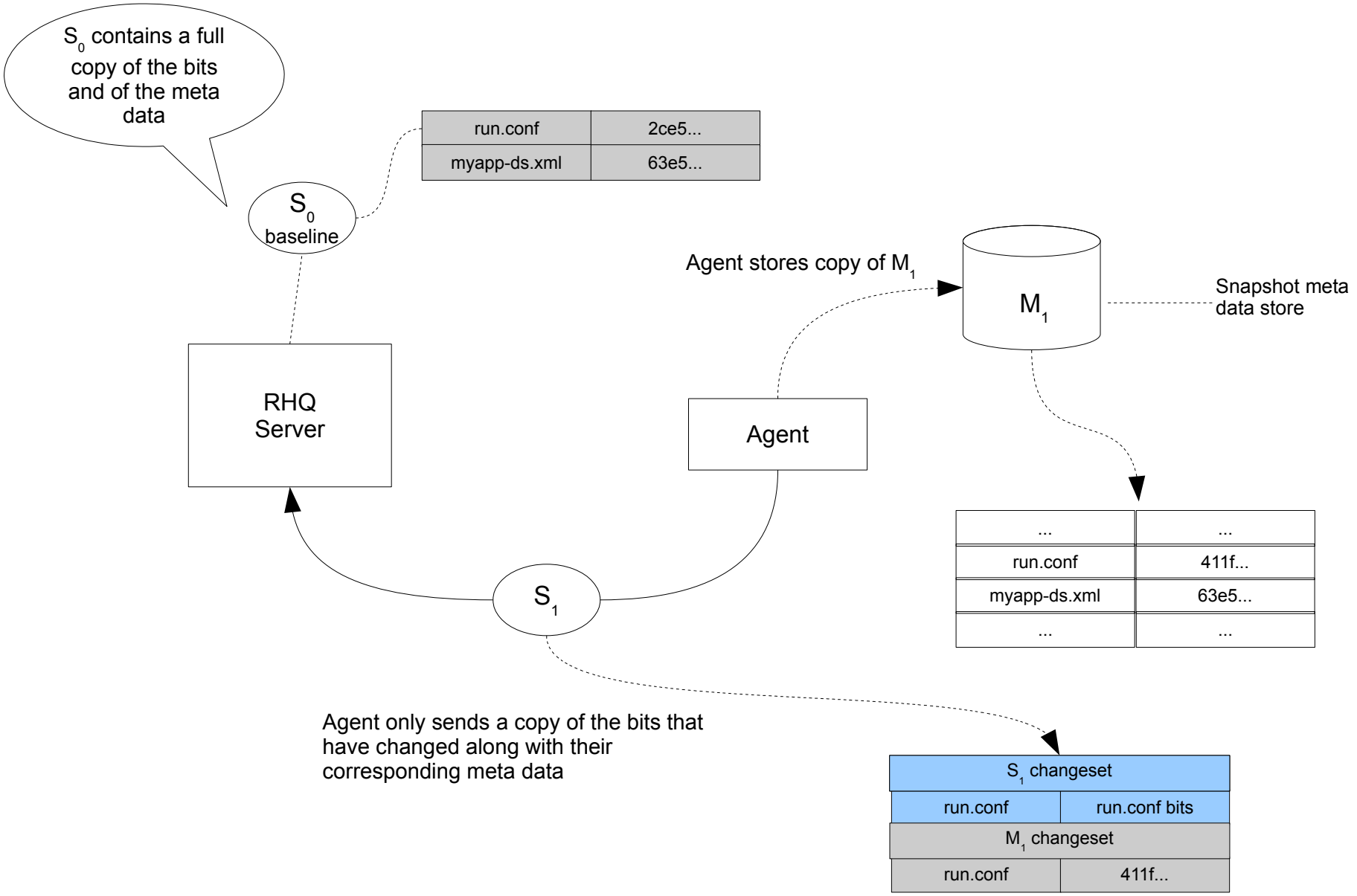
With this alert definition the server will fire an alert whenever it detects drift for the EAP server.

Agent sends initial snapshot, S_0

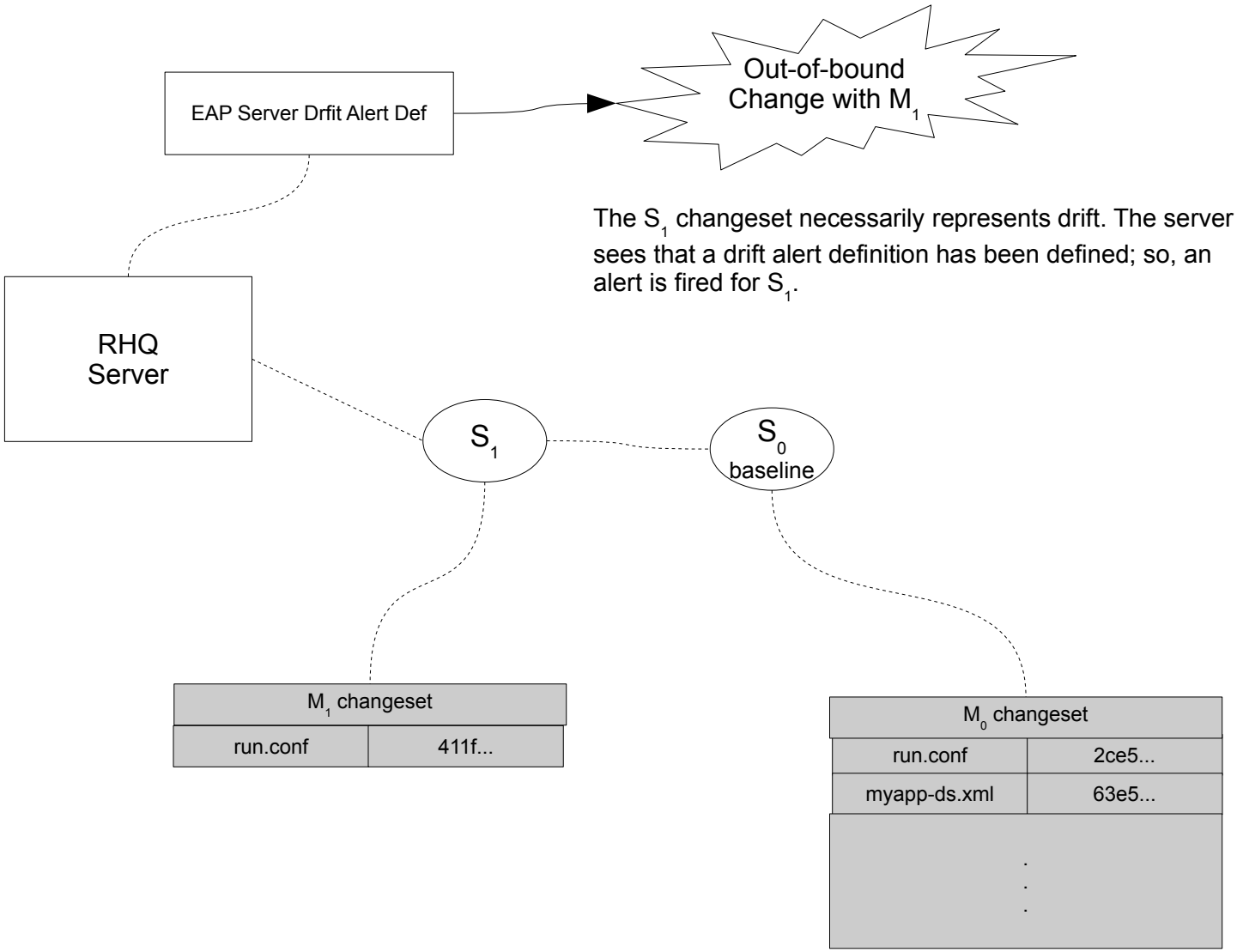


Agent sends initial snapshot S_0 to the server.
S_0 is a full copy with meta data M_0.

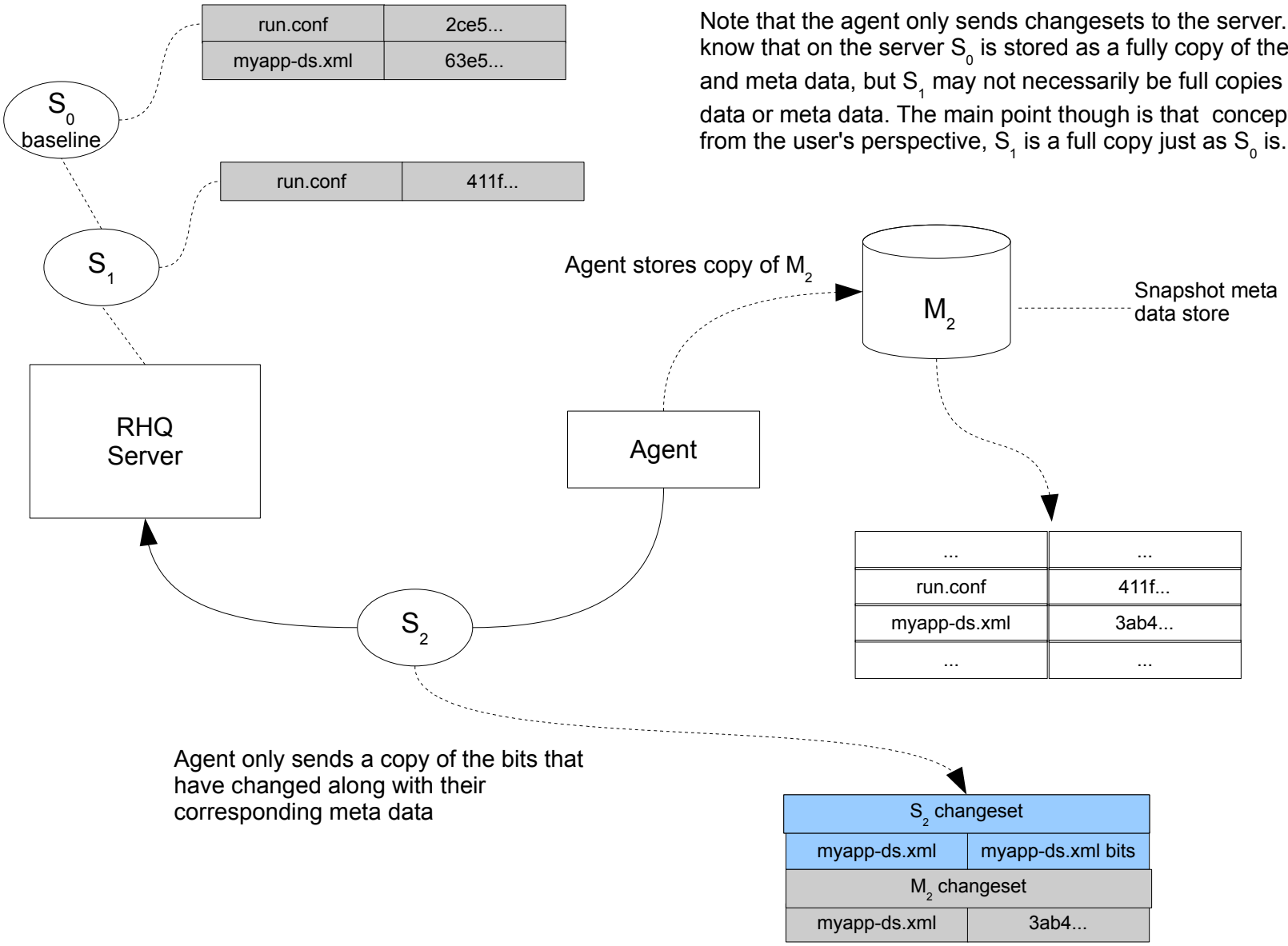
User lowers max heap size in run.conf



Server compares M_1 with M_0

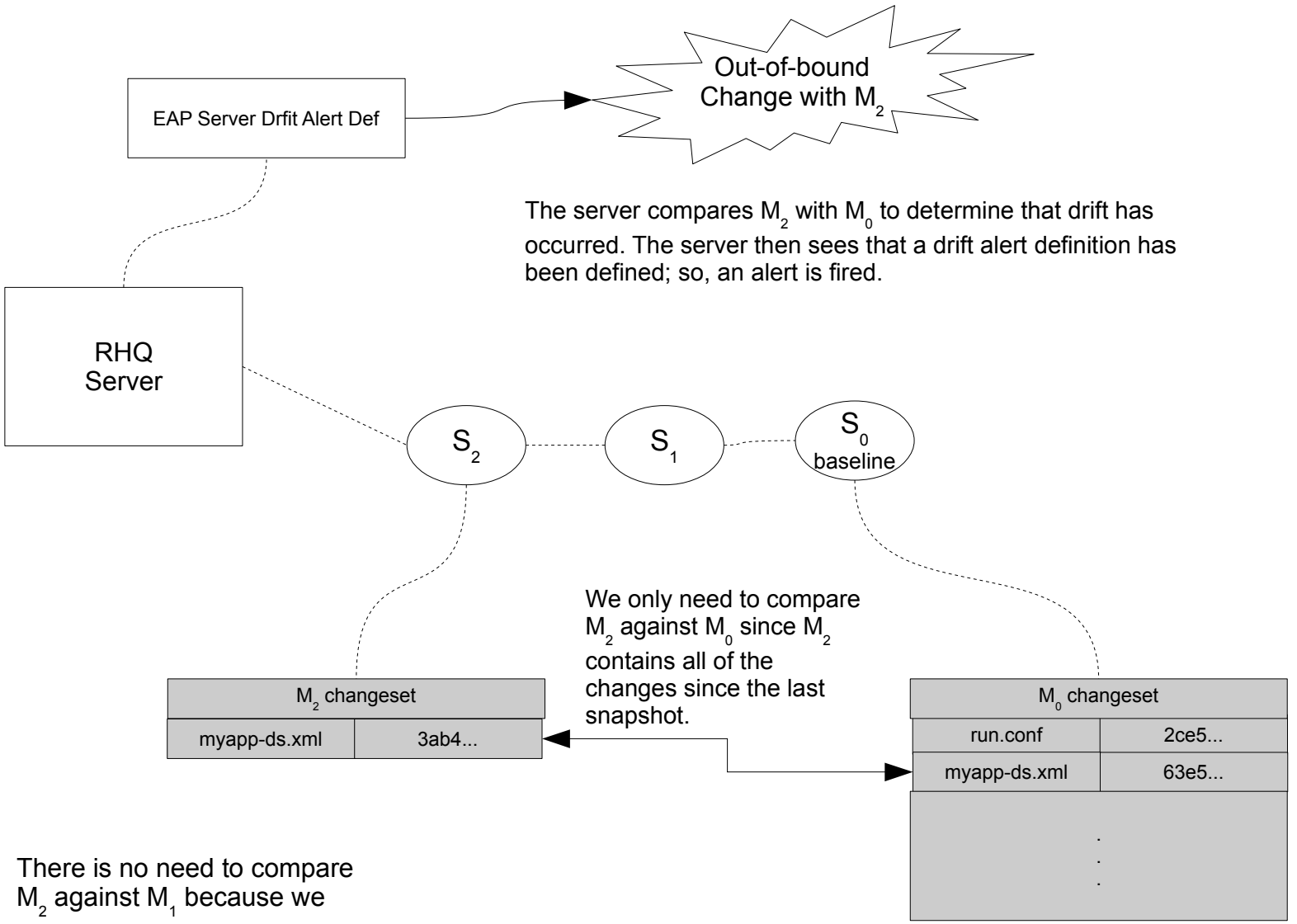


User modifies myapp-ds.xml



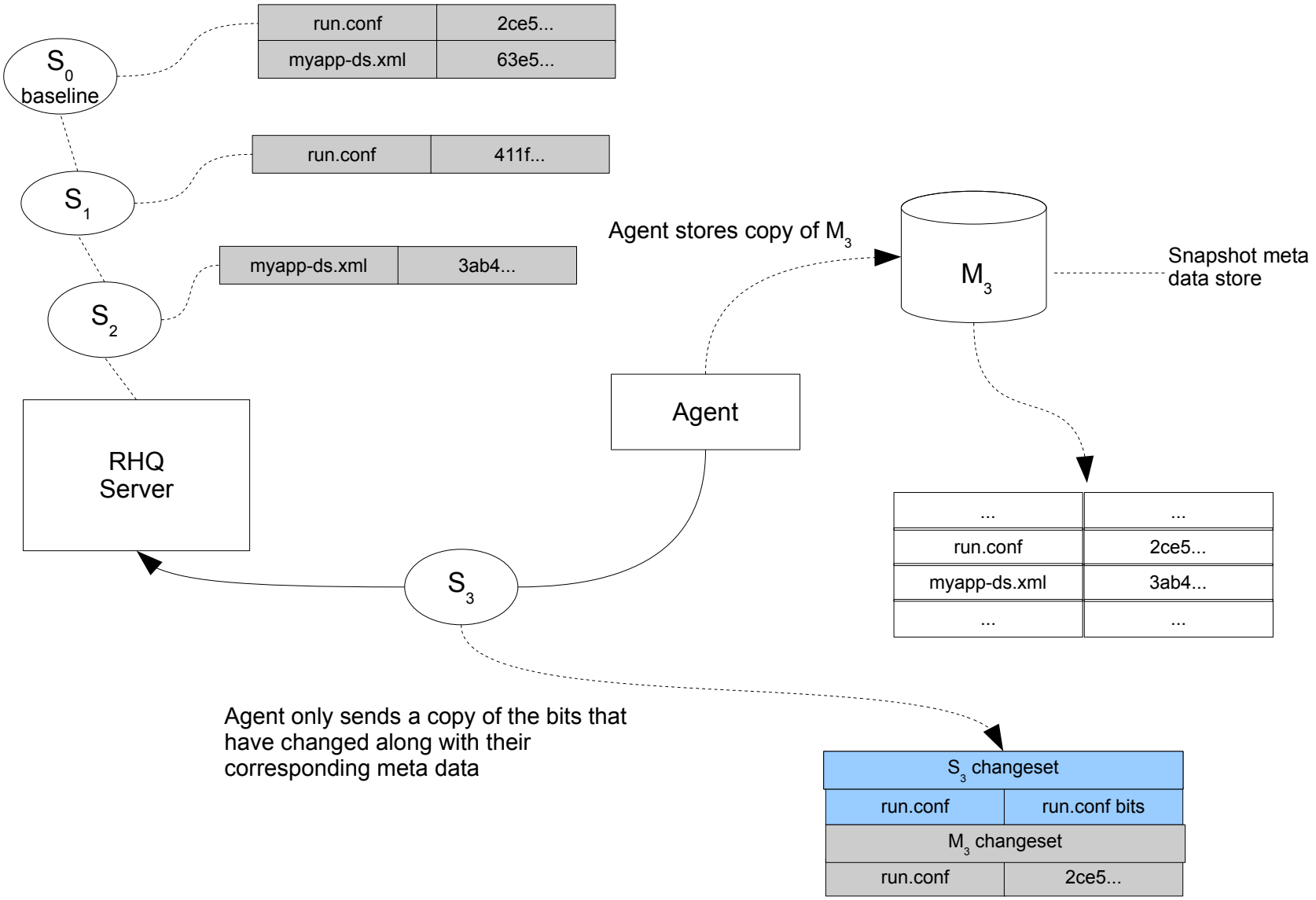
Note that the agent only sends changesets to the server. We know that on the server S_0 is stored as a fully copy of the data and meta data, but S_1 may not necessarily be full copies of the data or meta data. The main point though is that conceptually from the user's perspective, S_1 is a full copy just as S_0 is.

Server compares M_2 with M_0

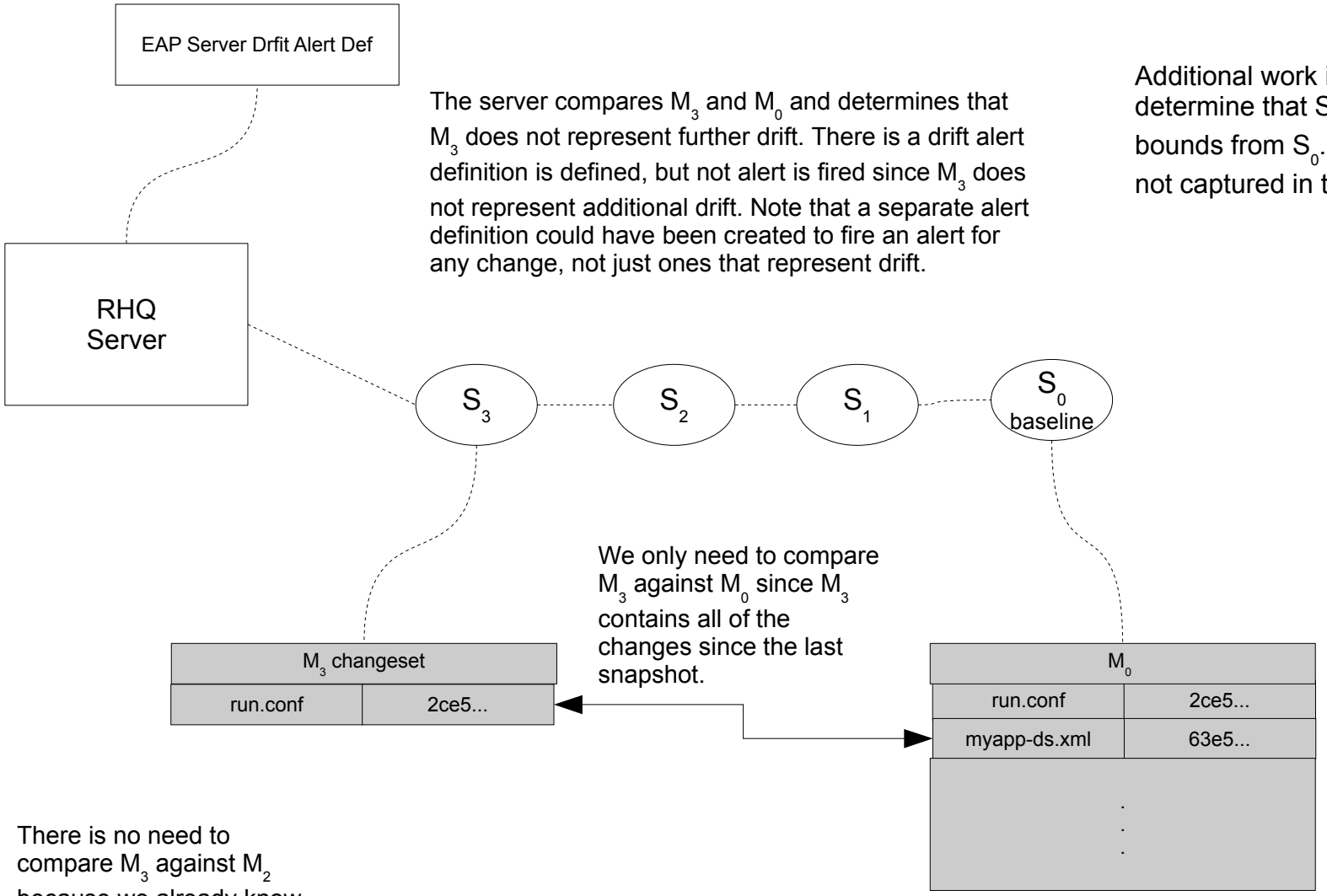


There is no need to compare M_2 against M_1 because we already know that M_2 represents the changes since M_1.

User increases max heap size to original value



Server compares M_3 with M_0



The server compares M_3 and M_0 and determines that M_3 does not represent further drift. There is a drift alert definition is defined, but not alert is fired since M_3 does not represent additional drift. Note that a separate alert definition could have been created to fire an alert for any change, not just ones that represent drift.

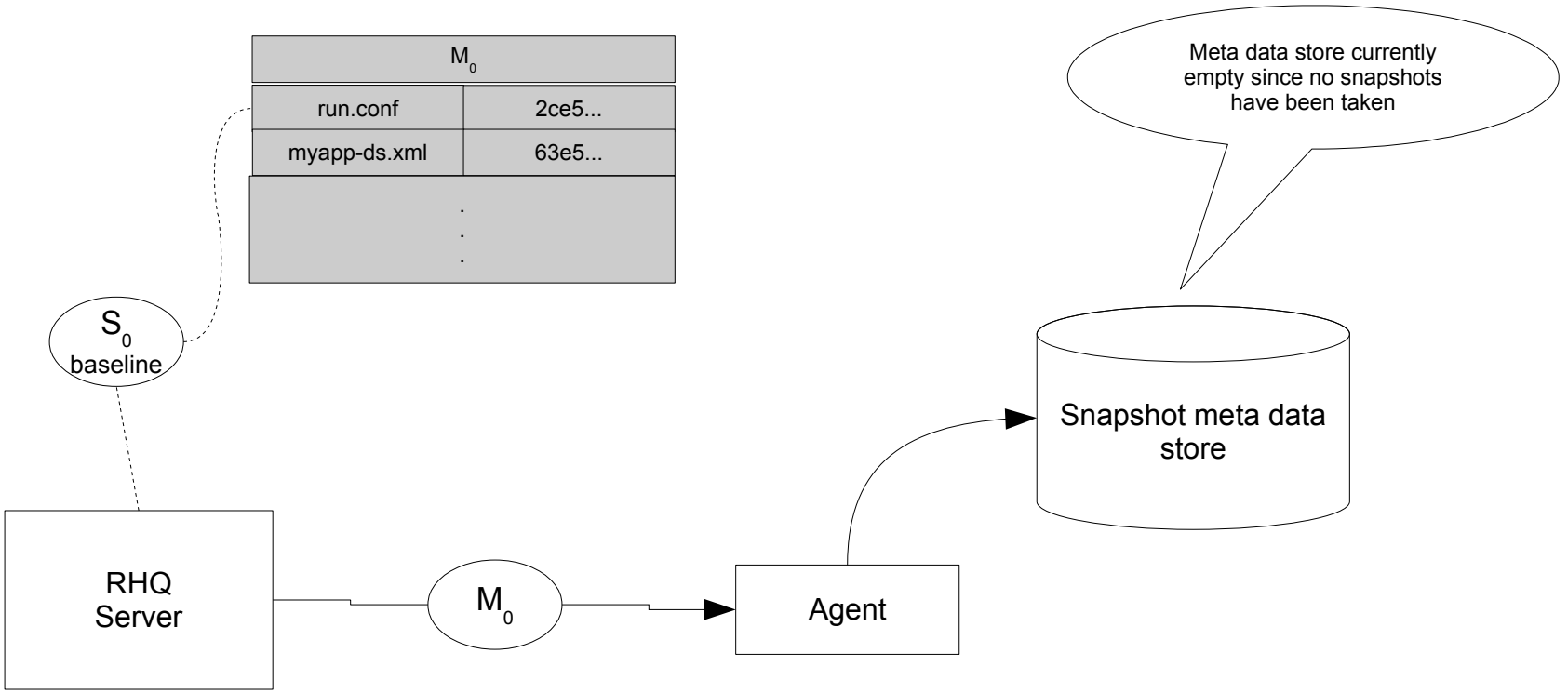
Additional work is required to determine that S_3 is still out of bounds from S_0. That work is not captured in this diagram.

We only need to compare M_3 against M_0 since M_3 contains all of the changes since the last snapshot.

There is no need to compare M_3 against M_2 because we already know that M_3 represents the changes since M_2.

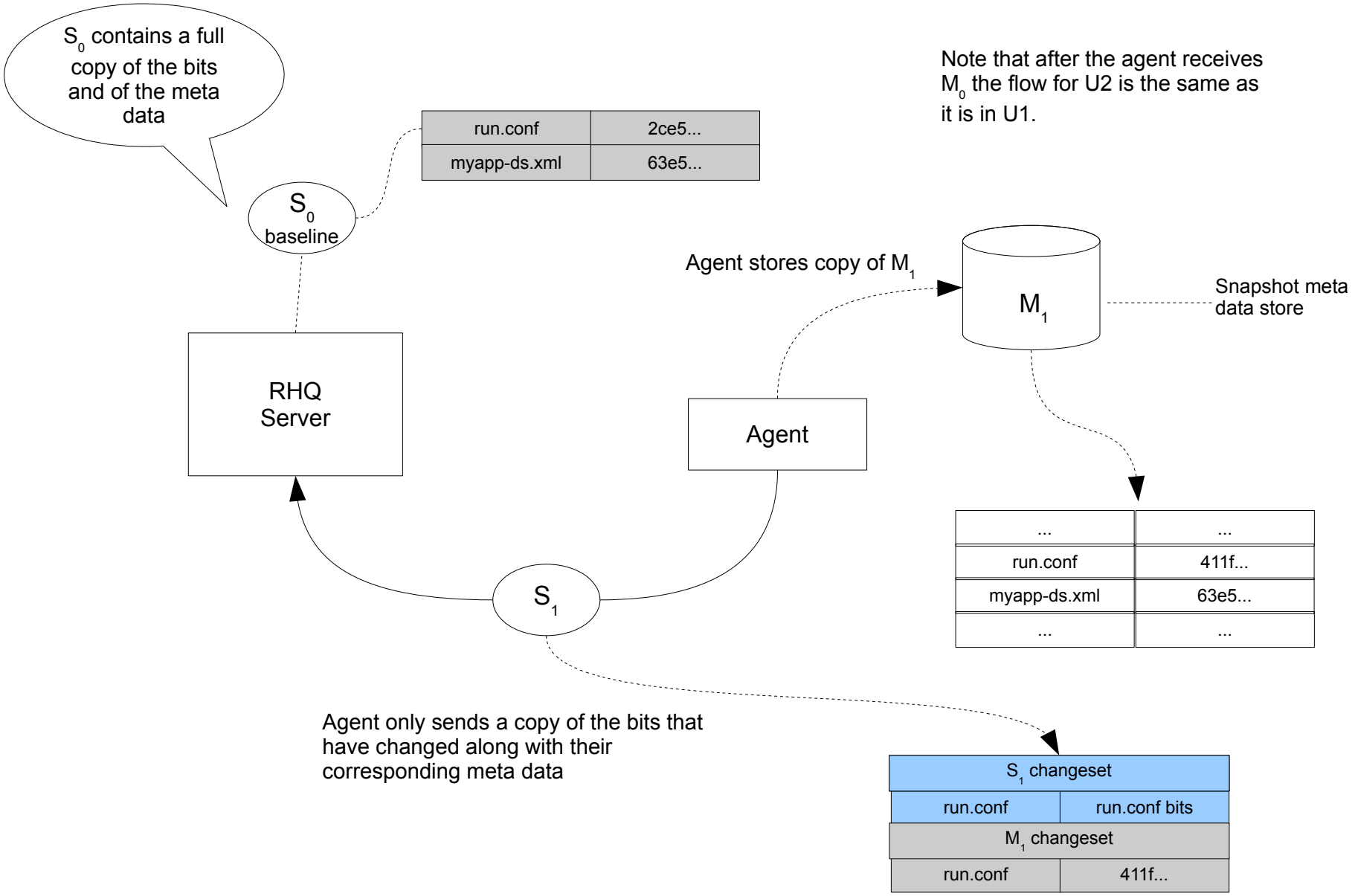
The current state of the EAP server represented by S_3 does not equal S_0; however, there is an equivalence between M_3 and M_0 since the changes in M_3 match corresponding values in M_0. We therefore say that M_3 is an in-bound change.

U2: Start drift monitoring for EAP server with pre-existing baseline snapshot

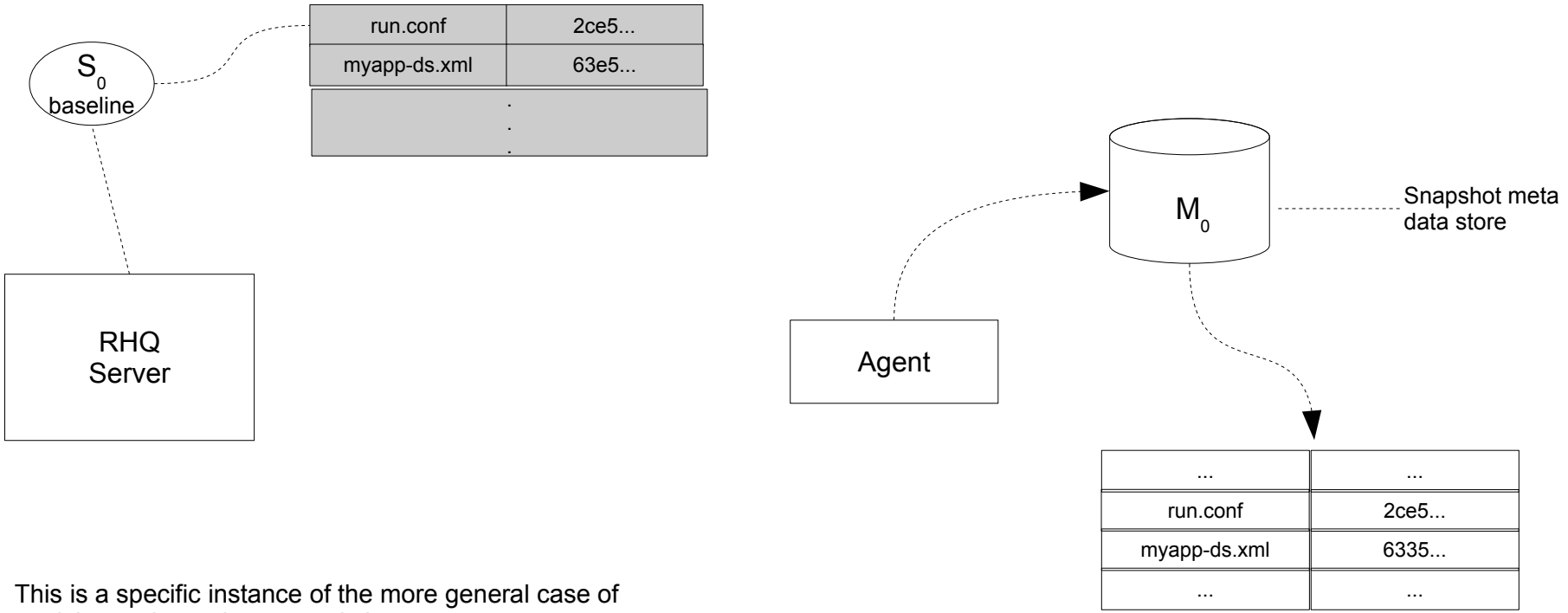


Request to start drift monitoring using pre-existing baseline S_0.
Server only sends M_0 for agent to use. It does not send a copy of the bits included in the snapshot.

User lowers max heap size in run.conf

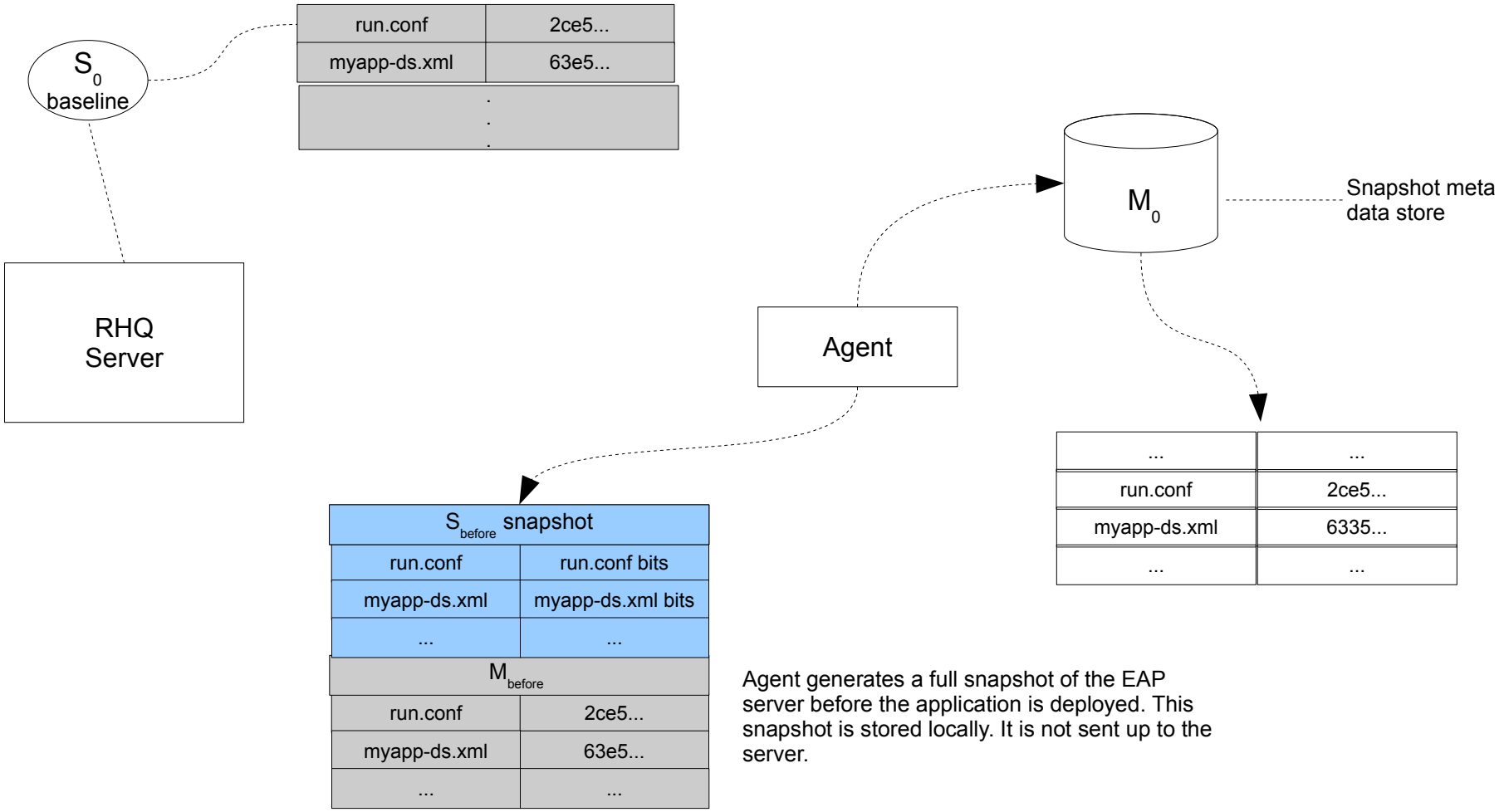


U3: Baseline snapshot exists and user wants to deploy application



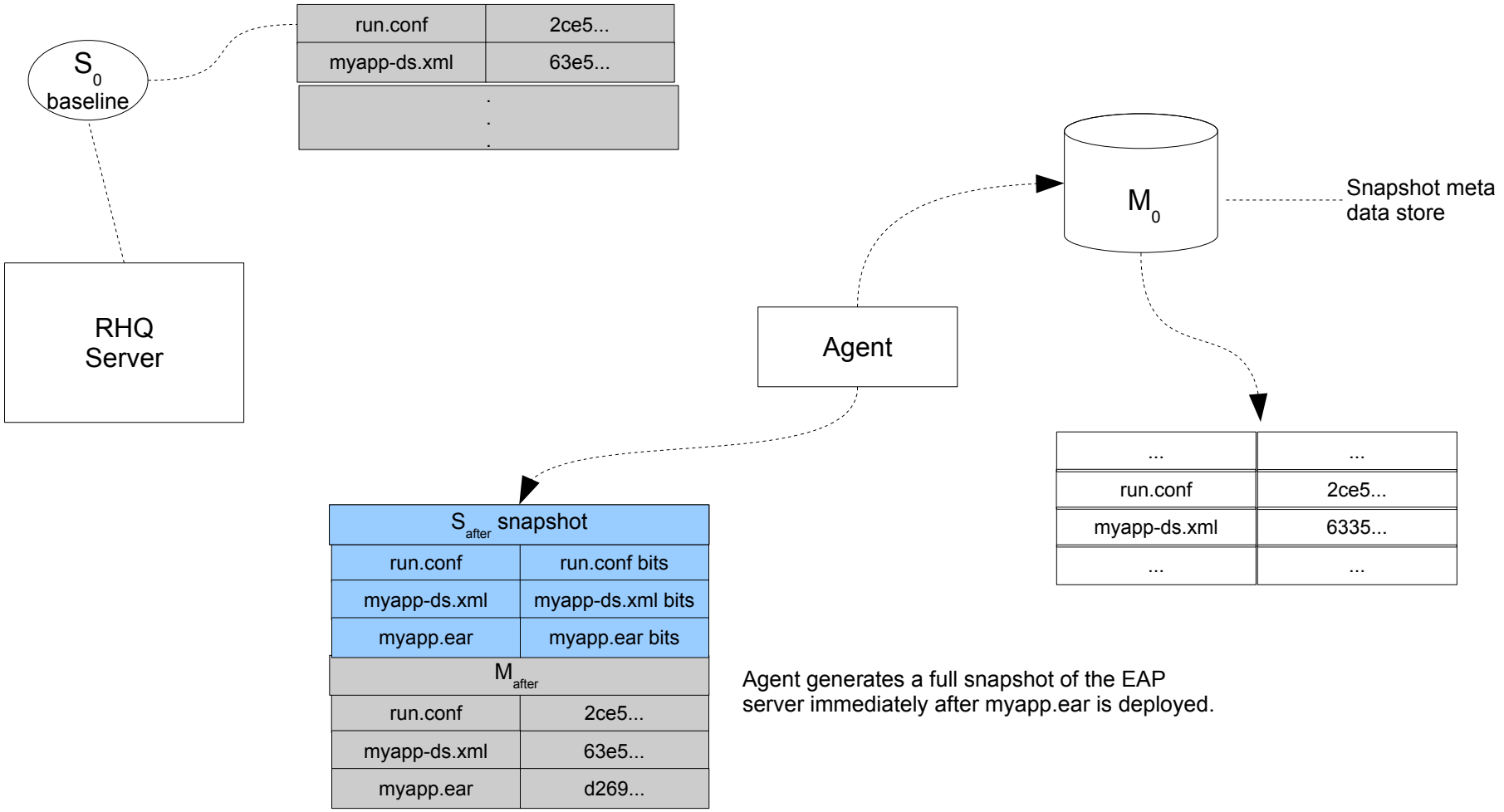
This is a specific instance of the more general case of applying a planned, expected change to a resource which has drift monitoring enabled. We want to apply the change while avoiding false drift detection. Applying planned configuration updates also falls under this use case.

Agent takes snapshot before application is deployed

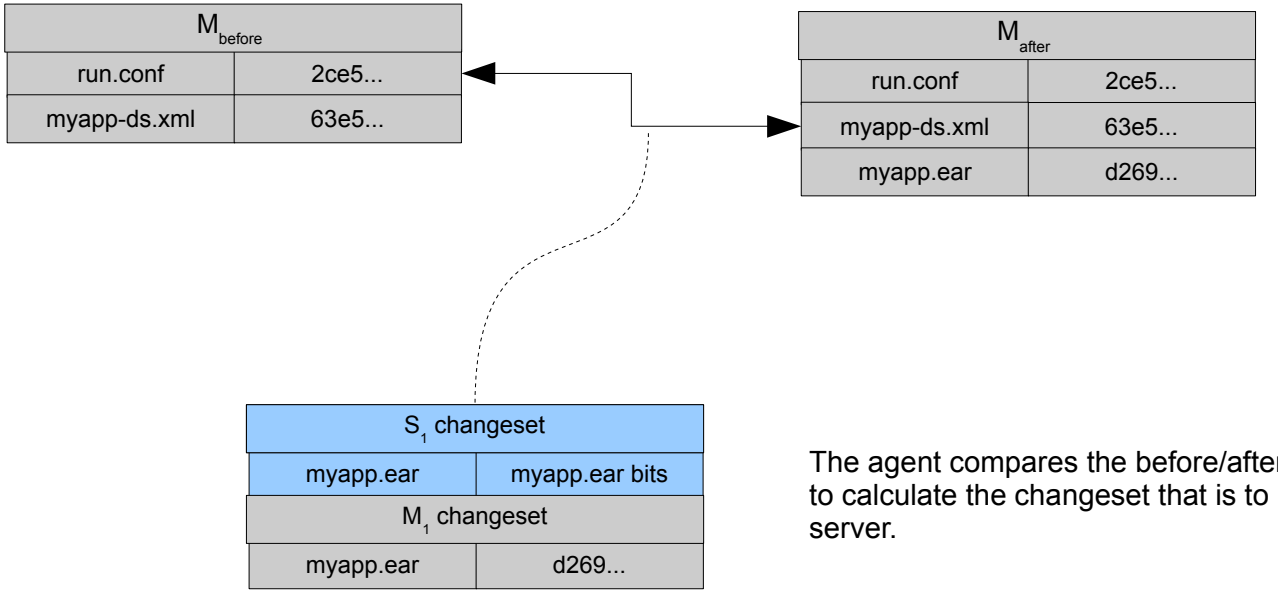


Agent generates a full snapshot of the EAP server before the application is deployed. This snapshot is stored locally. It is not sent up to the server.

Agent takes snapshot after application is deployed

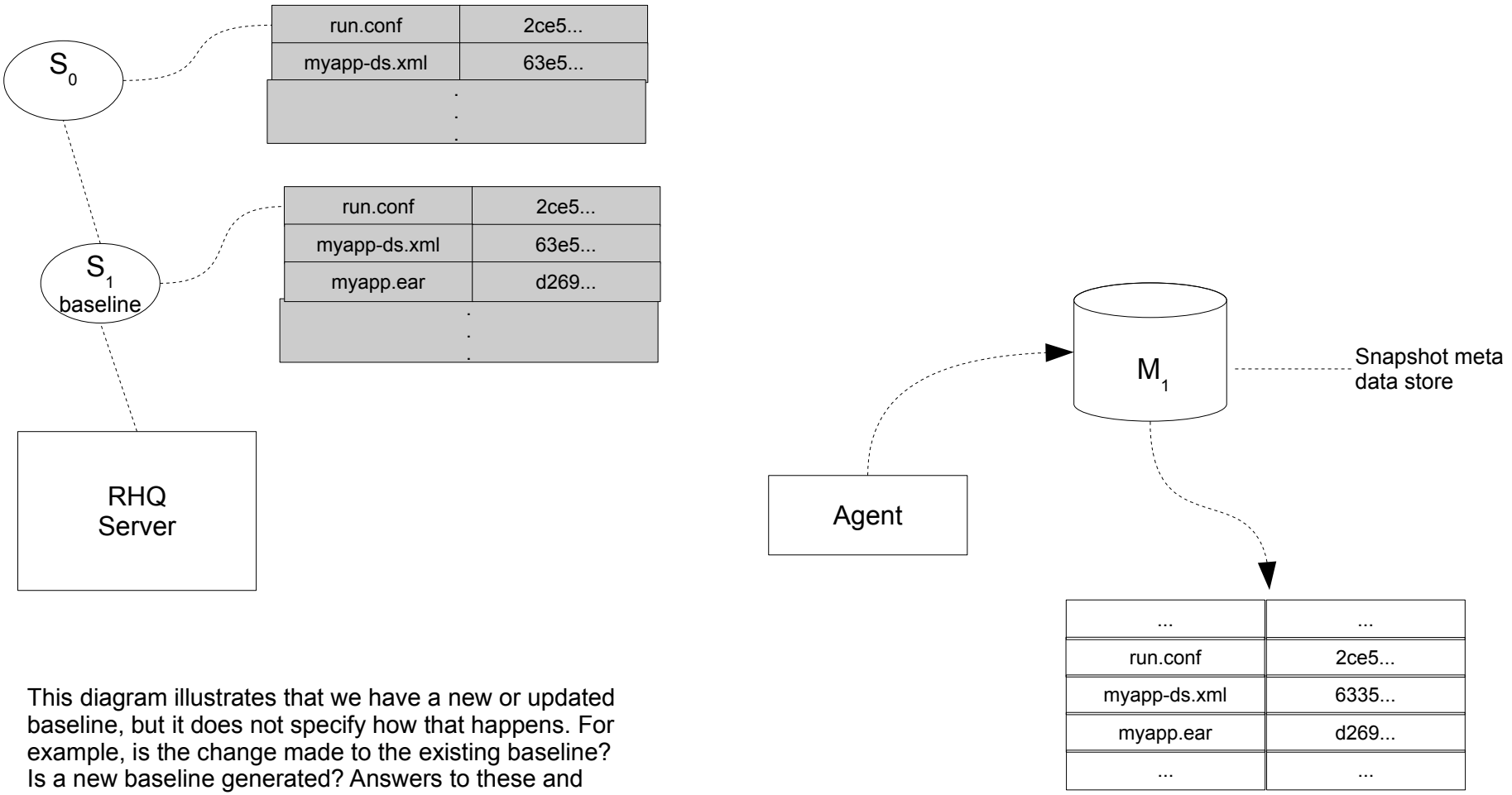


Agent compares before/after meta data



The agent compares the before/after meta data in order to calculate the changeset that is to be sent up to the server.

Server recalculates baseline



This diagram illustrates that we have a new or updated baseline, but it does not specify how that happens. For example, is the change made to the existing baseline? Is a new baseline generated? Answers to these and other questions are left as implementation details.