SSLSessionResumption

What's ssl session resumption

- When a client and server establish an SSL connection for the first time they need to establish a shared key called the master_secret. The master_secret is then used to create all the bulk encryption keys used to protect the traffic. The master_secret is almost invariably established using a public key algorithm such as EC, RSA or Diffie-Hellman (DH). Unfortunately, these algorithms are quite slow. In order to improve performance, SSL contains a "session resumption" feature that allows a client/server pair to skip this time consuming step if they have already established a master_secret in a previous connection. (from Eric Rescorla's article http://www.linuxjournal.com/article/5487)
- In addition, TLS extensions support a ticket based session resumption. On first TLS handshake the server returns an encrypted version of the session state. The client treats this as an opaque ticket and presents it on the next TLS handshake to the same server.

Challenges for TrafficServer

- For a single server session ID based resumption and ticket based resumption would work just fine. If there are parallel traffic servers deployed behind a virtual IP and a client may be directed to different real servers, then some coordination must occur to successful resume TLS sessions.
- For tickets, the parallel TrafficServers must share the same ticket encrypting keys. You can copy around files as discussed in the documentation, or use the TSSslTicketKeyUpdate API to programmatically share and update the ticket key.
- For session ID based resumptions, the TrafficServers must share session state. For 8.0 we added TLS Session Plugin API, so one could write a plugin to propagate session state between TrafficServer installations.
- A ssl_session_reuse plugin is in the experimental plugin tree. This plugin uses redis to communicate session state between traffic_server instances. It also implements an algorithm to coordinate ticket encrypting key creation.