Comprendre la connexion WebSocket pour Finesse

Table des matières

Introduction

Ce document décrit complètement la connexion WebSocket afin que, pendant le dépannage, les processus sous-jacents soient parfaitement compris.

Conditions préalables

Exigences

Aucune exigence spécifique n'est associée à ce document.

Composants Utilisé

Les informations contenues dans ce document sont basées sur les versions de matériel et de logiciel suivantes :

- Cisco Finesse
- UCCX

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. Si votre réseau est en ligne, assurez-vous de bien comprendre l’incidence possible des commandes.

Informations générales

Web Socket est une connexion permanente entre le client et le serveur.

Web Socket

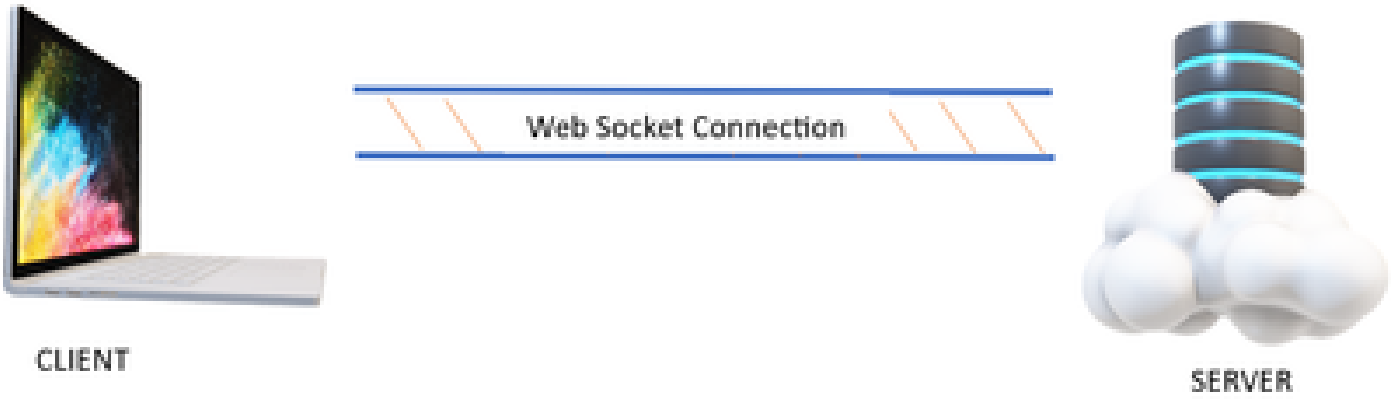
Qu'entend-on par connexion permanente ?

Cela signifie qu'une fois la connexion établie entre le client et le serveur, le client et le serveur peuvent envoyer et/ou recevoir des données à tout moment.

Il s'agit d'une connexion bidirectionnelle bidirectionnelle en mode bidirectionnel.

Le serveur n'a pas besoin d'attendre la requête du client pour repousser les données.

De même, le client n'a pas besoin de créer une nouvelle connexion à chaque fois pour envoyer de nouvelles données au serveur.

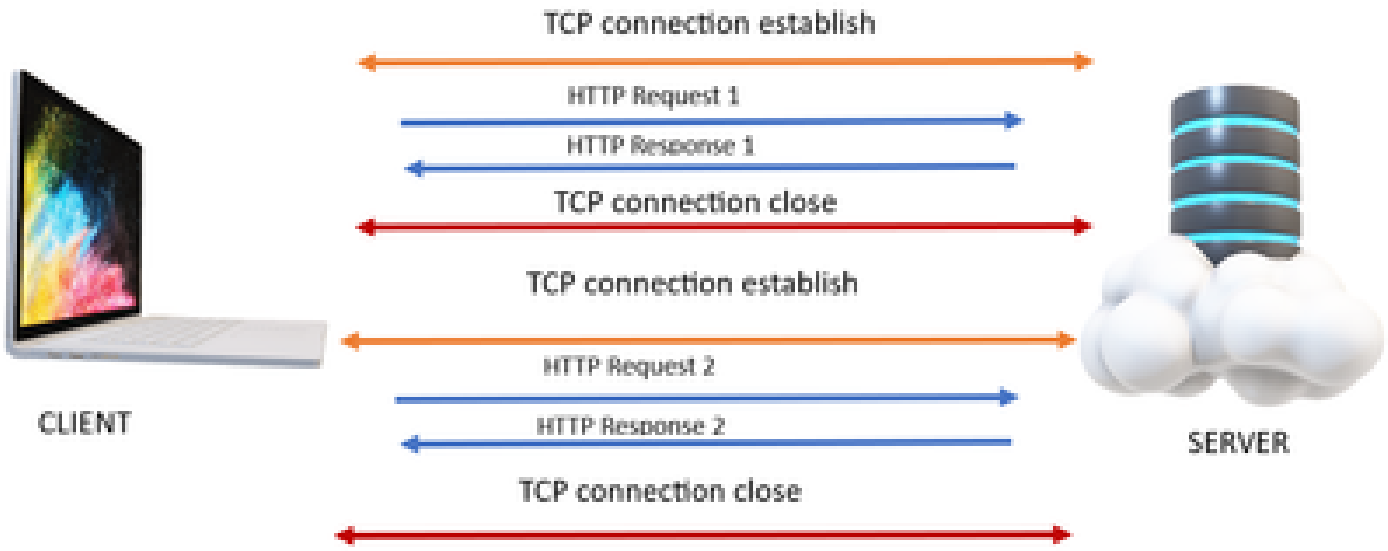


La connexion Web Socket est principalement utilisée dans les applications où des mises à jour de données en temps réel sont nécessaires.

Par exemple, les applications de trading boursier, les applications de messagerie et, dans notre cas, Cisco Finesse.

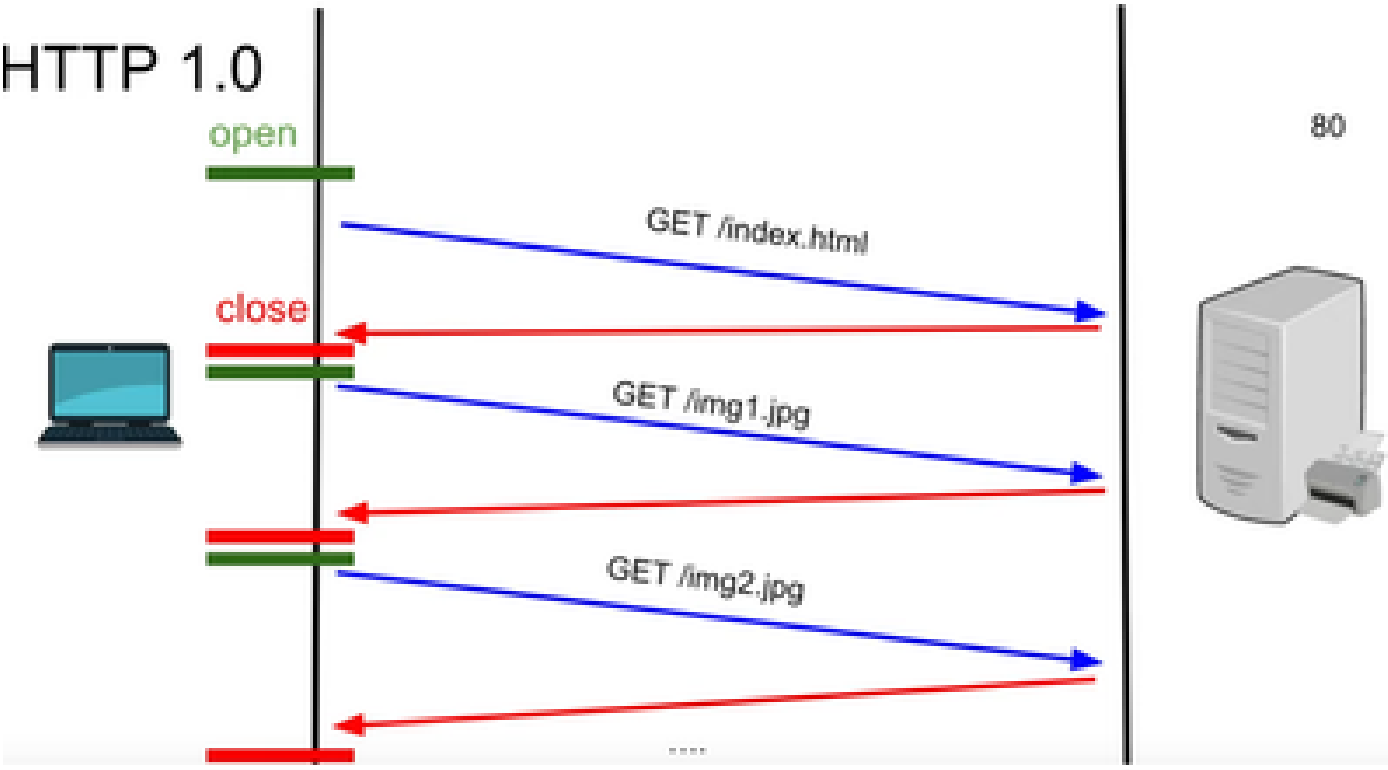
Comment WebSockets fonctionne-t-il ?

Considérons :

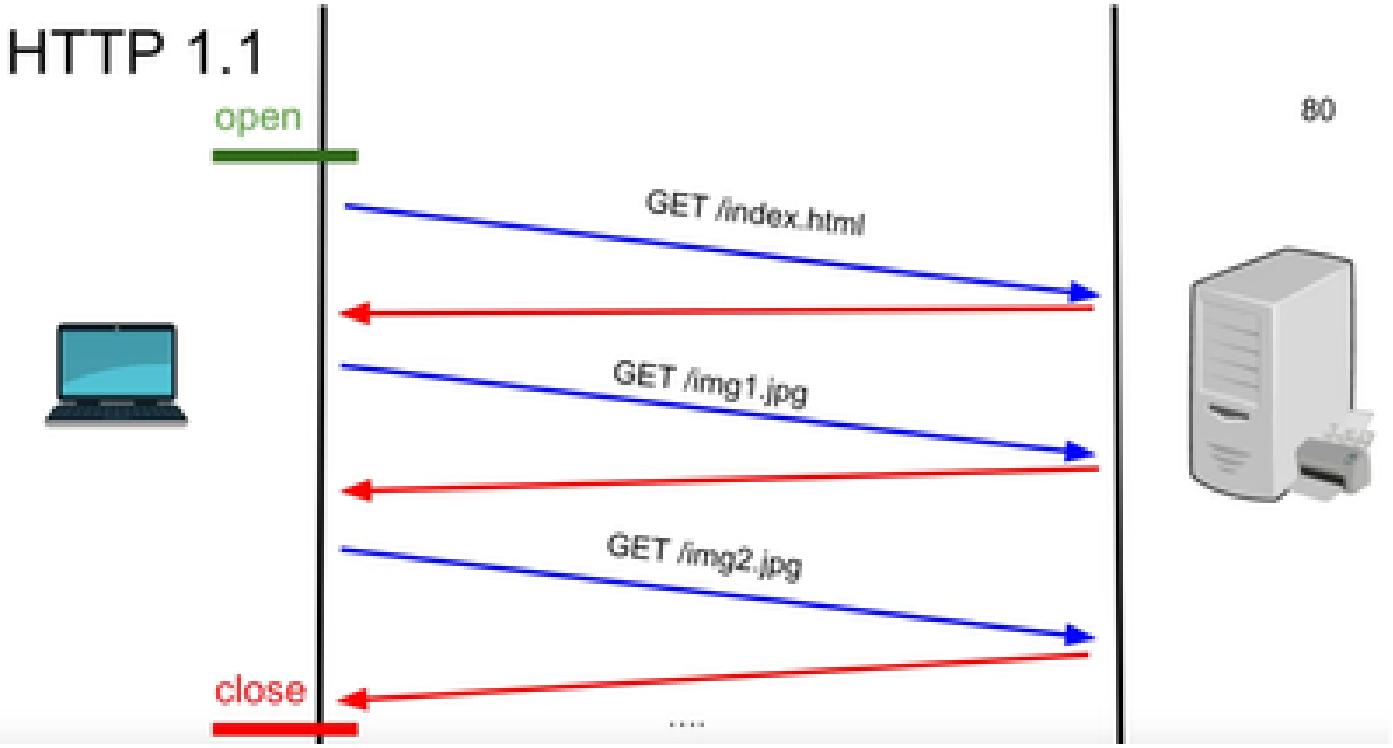


HTTP

1. La connexion TCP (connexion en trois étapes) a lieu.
2. Le client envoie ensuite une requête HTTP.
3. Le serveur envoie une réponse HTTP.
4. Après un cycle de réponse à la requête, la connexion TCP se ferme.
5. Dans le cas d'une nouvelle requête HTTP, la connexion TCP établit d'abord.



HTTP 1.0 - Après chaque réponse de requête, la connexion TCP redémarre pour une autre réponse de requête HTTP.



HTTP 1.1 - Cette connexion a fonctionné car vous pouviez envoyer et recevoir des données, puis fermer la connexion.

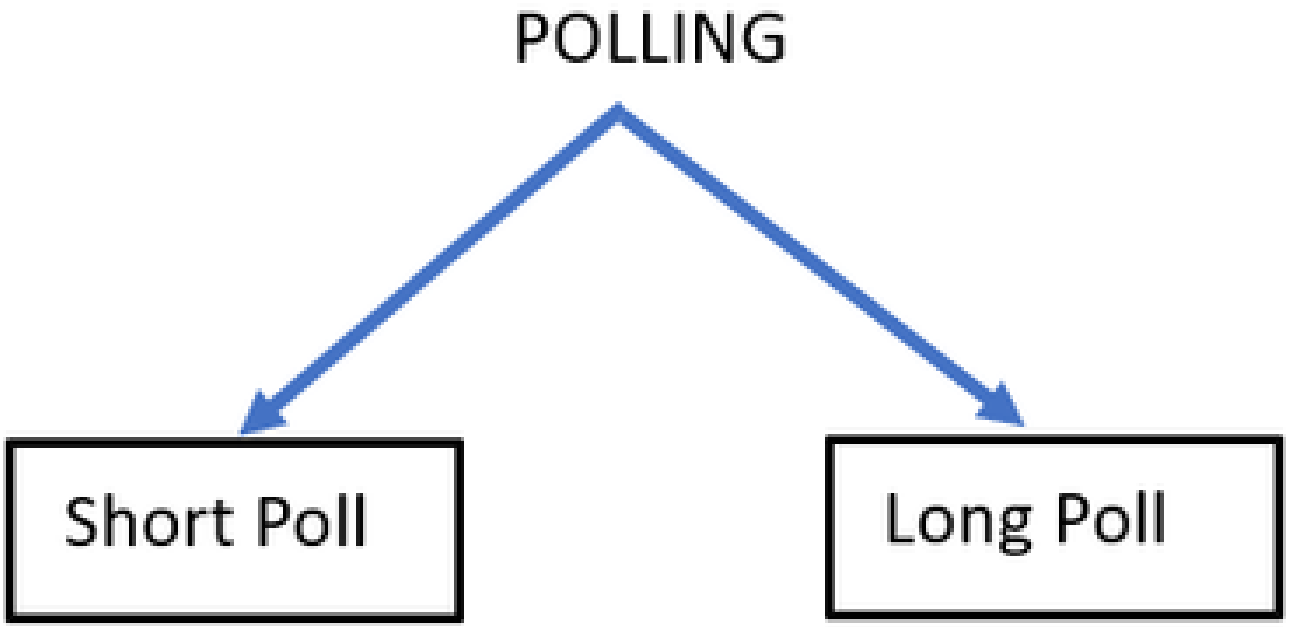
Encore une fois, cela n'était pas adapté aux applications en temps réel, car le serveur peut envoyer certaines données même lorsque le client ne le demande pas. Par conséquent, ce modèle n'est pas efficace.

Problème avec HTTP

Le problème commence avec les systèmes en temps réel.

Pour un site web qui nécessite des mises à jour en temps réel, il est très difficile d'envoyer des requêtes HTTP à chaque fois pour obtenir une mise à jour du serveur et il utilise beaucoup de bande passante et provoque une surcharge.

Pour résoudre ce problème, un mécanisme de HTTP appelé Interrogation est utilisé.



Sondage court : le est mis en oeuvre lorsqu'un compteur court fixe est défini pour les demandes et les réponses. Par exemple, 50 secondes ou 1 seconde selon l'implémentation.

S'il n'y a pas de mise à jour de l'autre côté, alors vous pouvez obtenir des réponses vides dans ce délai qui peut gaspiller des ressources.

Sondage long - Il surmonte en quelque sorte le sondage court, mais a encore un temps fixe pour attendre une réponse.

S'il n'y a pas de réponse dans ce délai qui est relativement plus long que le bref sondage, mais qu'il est toujours fixé, alors demandez à nouveau des délais d'attente.

Par conséquent, le sondage n'est pas la meilleure façon de surmonter ce problème.

Pour cela, l'autre méthode à utiliser est appelée SSE.

SE

Événements envoyés par le serveur

Dans ce cas, il existe une connexion unidirectionnelle entre le serveur et le client par laquelle le serveur peut envoyer les données au client à tout moment.

La chose à noter ici est qu'il s'agit d'une connexion unidirectionnelle, ce qui signifie que seul le serveur peut envoyer les données au client et non l'inverse.

Exemple d'utilisation : notifications ou mises à jour groupées d'un serveur vers un client. Par exemple, actualités en direct, Instagram en direct, etc.

Cette méthode n'est pas très efficace pour les applications impliquant des mises à jour et des

messages en temps réel.

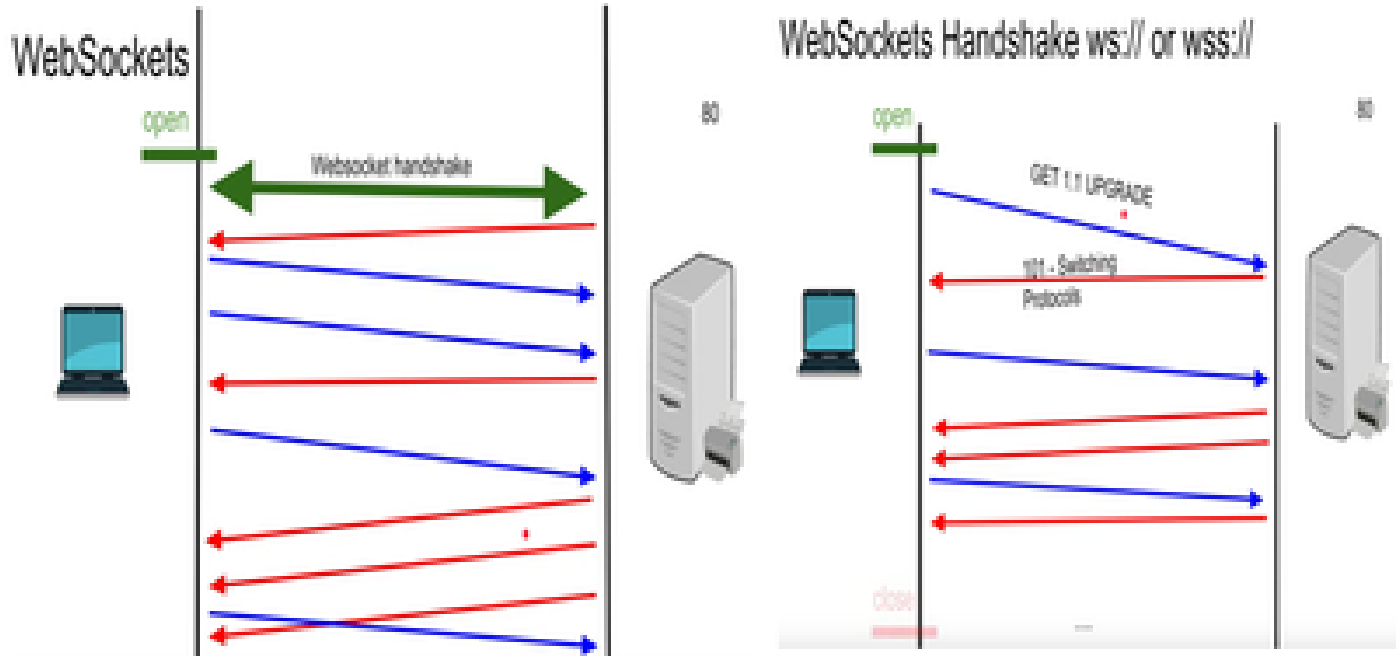
La connexion Web Socket est une connexion bidirectionnelle bidirectionnelle en mode bidirectionnel.

Il peut s'agir d'un appel téléphonique entre un serveur et un client au cours duquel n'importe quel interlocuteur peut parler à l'autre à tout moment.

Actions WebSocket

1. Pour établir une connexion WebSocket, le client envoie une requête de connexion HTTP avec un en-tête mis à niveau ou mis à jour.
    1. Cela signifie que le client dit au serveur qu'à l'heure actuelle, c'est sur HTTP, mais qu'à partir de maintenant, il se déplace sur la connexion de la websocket.
2. Le serveur répond alors par la réponse HTTP 101, ce qui signifie que le site commute la réponse de protocole.
3. Ensuite, la connexion de la websocket est établie.

Désormais, le serveur et le client peuvent utiliser cette même connexion pour se transférer des données à tout moment.



Débogages WebSocket

Si à ce stade vous vous connectez au client Finesse et voyez les débogages du réseau, il s'affiche comme suit :

MÉTHODE - GET

Domaine - NOM DU SERVEUR

FICHIER - /WS/

INITIATEUR - Openfire.js - WebSocket

Examen de la demande et de la réponse :

Demande

GET

Schéma : wss

hôte : uccxpub.prabhat.com:8445

nom du fichier : /ws/

Adresse : IP du serveur uccx

État : 101

Commutation Protocoles

VersionHTTP/1.1

EN-TÊTE DE RÉPONSE

Connexion : mise à niveau

Mise à niveau : WebSocket

Request – open

Response

PLAIN

http://jabber.org/protocol/caps" hash="sha-1" node="
https://www.igniterealtime.org/projects/openfire/" ver="k3mOuil8afx3OTZxYy6yxLmFsok="/>

Request - auth

YWRtaW5pc3RyYXRvckB1Y2N4cHViLnByYWJoYXQuY29tAGFkbWluaXN0cmF0b3IAMTIzNA==

Response

Request – XMPP Bind Bind request to Bind the resource which in this case is desktop with a jabber id

desktop

Response – XMPP Bind where User ID is given a jabber id

administrator@uccxpub.prabhat.com/desktop

administrator@uccxpub.prabhat.com/desktop

Presence request

Presence response

http://jabber.org/protocol/caps" hash="sha-1" node="
http://www.igniterealtime.org/projects/smack" ver="NfJ3flI83zSdUDzCEICtbypursw=">

http://jabber.org/protocol/caps" hash="sha-1" node="
http://www.igniterealtime.org/projects/smack" ver="NfJ3flI83zSdUDzCEICtbypursw=">

PUBSUB request – Requesting to subscribe the user to the pubsub node so that all the events on the user are monitored.

Response – user subscribed.

http://jabber.org/protocol/pubsub">

PUBSUB request – Requesting to subscribe the Team to the pubsub node so that all the events on the team are monitored.

Response – Team subscribed

http://jabber.org/protocol/pubsub">

Informations connexes

- Assistance technique de Cisco et téléchargements

À propos de cette traduction
Cisco a traduit ce document en traduction automatisée vérifiée par une personne dans le cadre d'un service mondial permettant à nos utilisateurs d'obtenir le contenu d'assistance dans leur propre langue.

Il convient cependant de noter que même la meilleure traduction automatisée ne sera pas aussi précise que celle fournie par un traducteur professionnel.