CORS全文為(Cross-Origin Resource Sharing),翻成中文就是跨網域資源共享，也就是你可以用我資源我也可以用你的。是一個瀏覽器做跨網域連線的方式，此機制提供了網頁伺服器跨網域的存取控制，增加跨網域資料傳輸的安全性。透過HTTP header 的設定，可以規範瀏覽器在進行跨網域連線時可以存取的資料權限與範圍，包括哪些來源可以存取，或是哪些 HTTP verb, header 的 request 可以存取。

因為安全性的考量，瀏覽器預設都會限制網頁做跨網域的連線。但如果要提供資料存取的服務給其它人使用，就必須要開放對應的API給其它人連線。而 CORS 就是一個瀏覽器做跨網域連線的時要遵守的規範。雖然Same Origin Policy不錯,因為他防止了一些惡意的 script 攻擊,但總不會每一個跨網域都是惡意的;也不可能一間公司擁有所有的資源，有時還是必須串接第三方資源，例如 Facebook API、Google Map、政府釋出的公開 API 等，因此我們會使用CORS。

而在實作的方面可以說是相當容易，它其實只是 HTTP-Header 而已，這些設定基本上都是在後端，所以前端只需知道概念跟怎麼看 Responce Header 即可。當前端用 fetch 或 XMLHttpRequest 要存取資源時,在 request 之前都會先發送 preflight request 確定 server 端有設定正確的相關 Http-Header，若檢查通過，才會實際發出 requeest。

而CORS的流程是當一個支援 CORS 瀏覽器在網頁送出一個 request 時,會做以下的動作，瀏覽器根據送出 request 的 HTTP verb 與 header，判斷這個 request 是一個簡單請求(simple request)或是非簡單請求(判斷的細節可參考MDN - HTTP access control (CORS) - Simple requests)。如果是一個簡單請求，則直接送出 request。 如果是一個非簡單請求的 request，則進行 CORS preflight。先對伺服器送出一個 verb 為 OPTION 的 preflight request，它會帶有特定的 header 告訴伺服器接下來的 request 需要哪些跨網域連線的權限。當伺服器收到 preflight 後，就會回傳帶有特定 header 的 response 給瀏覽器，告訴它有哪些權限是允許的。取得伺服器的 response 後，如果符合連線權限，就會送出真正的 request。如果發現權限不符，就會出現錯誤訊息而中斷送出 request 的步驟。

如果要做跨網域連線，伺服器端在送出 preflight response 時，就必須要帶有特定的 header。但非常不建議這樣做,因為這樣的設定會讓所有的 response 都開放跨網域連線，顯然是個不安全的行為。比較好的做法是使用 rack-cors 這個 gem 來處理 CORS，它的用法也很簡單，只要在 config 中加入相關的設定就可以了。