

圖 1 ：Prometheus官方文檔架構圖

Retrieval system主要的功能是以pull拉取的方式取得指定的監控目標target資訊後儲存的模組，定義在Scrape模組中，包含三個檔案manager.go、scrape.go、target.go()(圖 2)。每一個目標target有一個與之對應的循環loop，每個loop內部執行 Http Get請求拉取數據。每一個job有一個與之對應的scrape pool，通過一些控制參數，執行周期性數據採集及結束等運作。

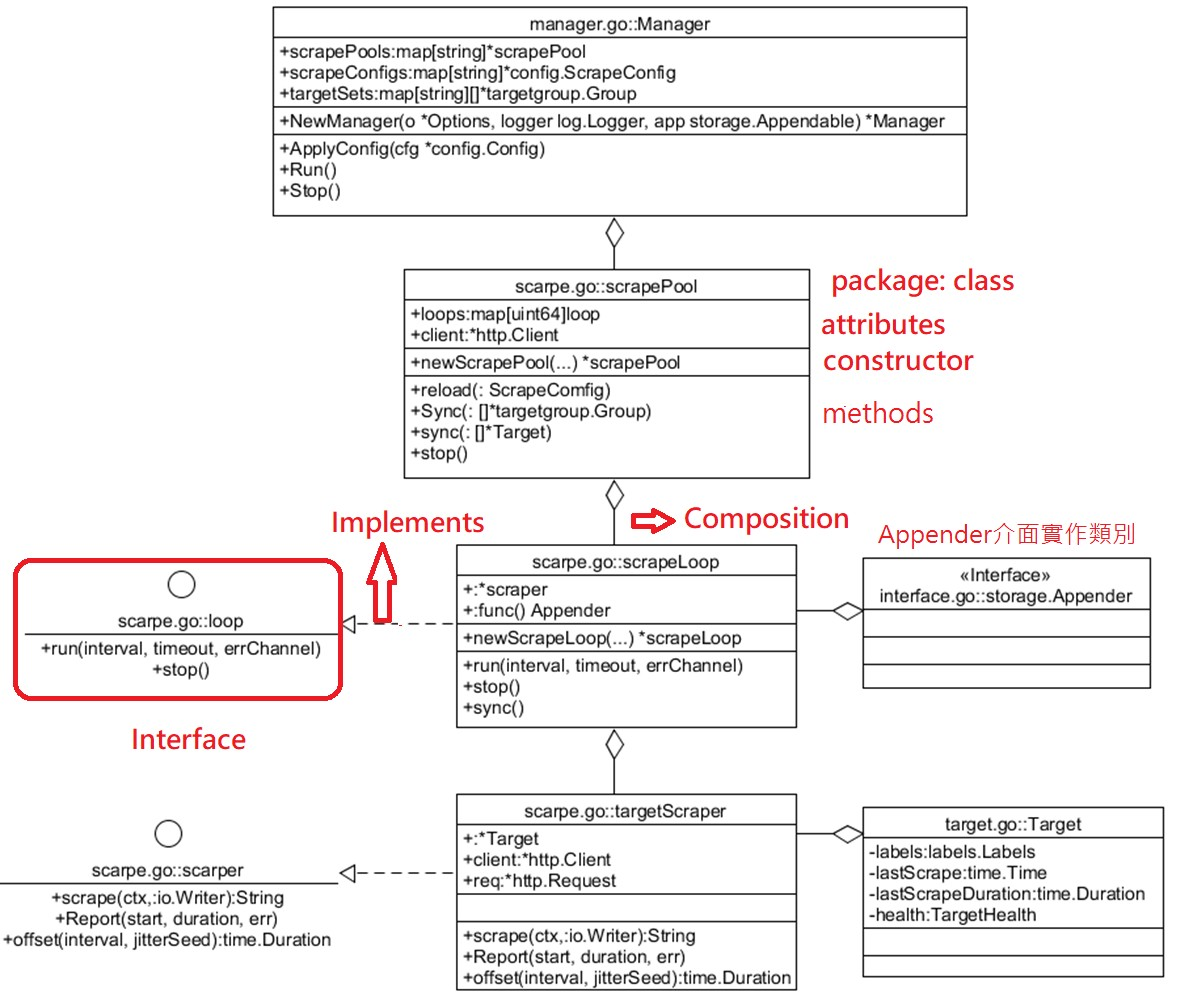


圖 2：Retrieval System Class Diagram

manager.go包含

Manager 類別:由discovery manager 取得數據採集目標，記錄在scrape pools之中，並負責啟動及停止週期性的資料採集任務。

Manager類別主要的屬性有管理目標資料採集的scrapePools，採集設定scrapeConfig，目標集合targetSets。

NewManager() 是Manager的建構子

ApplyConfig()方法使用新的設定cfg 重置管理的目標提供者和作業配置，Run()方法接收並儲存目標集數據更新，並觸發抓取循環來重新加載reloader()，重新加載為背景執行，不會阻止接收目標數據更新，reload()則為每個job建立對應的scrape pool。

Stop()方法會取消所有正在運行的抓取池和區段，直到完全結束。

scrape.go包含

scrapePool類別：管理目標集合的資料採集，主要屬性有執行資料拉取的loops以及Http 端點的client。

newScrapePool()是scrapePool建構子。

reload()方法使用給定的抓取配置重新加載抓取池，目標狀態被保留，但所有抓取循環都使用新的抓取配置重新啟動。

Sync(tgs []\*targetgroup.Group)方法將目標群組轉換為實際的抓取目標，將當前運行的抓取器與結果集同步，並傳回所有抓取和刪除的目標。

sync(targets []\*Target) 方法對一個目標列表進行同步處裡，新目標啟動抓取循環，失效的目標則停止抓取循環。

stop()方法終止所有抓取循環。

scrapeLoop類別：實作loop介面的資料拉取循環物件，可以進行啟動run()及停止stop()呼叫，停止後不得重複使用。主要屬性有拉取資料的Scraper以及附加器Appender。

targetScraper類別：實作scraper介面的目標拉取物件，以Http Get執行請求，呼叫scarpe()抓取資料，Report()回報資料，offset()時間偏移量設置。

target.go包含Target 類別代表單一HTTP或HTTPS 的端點。

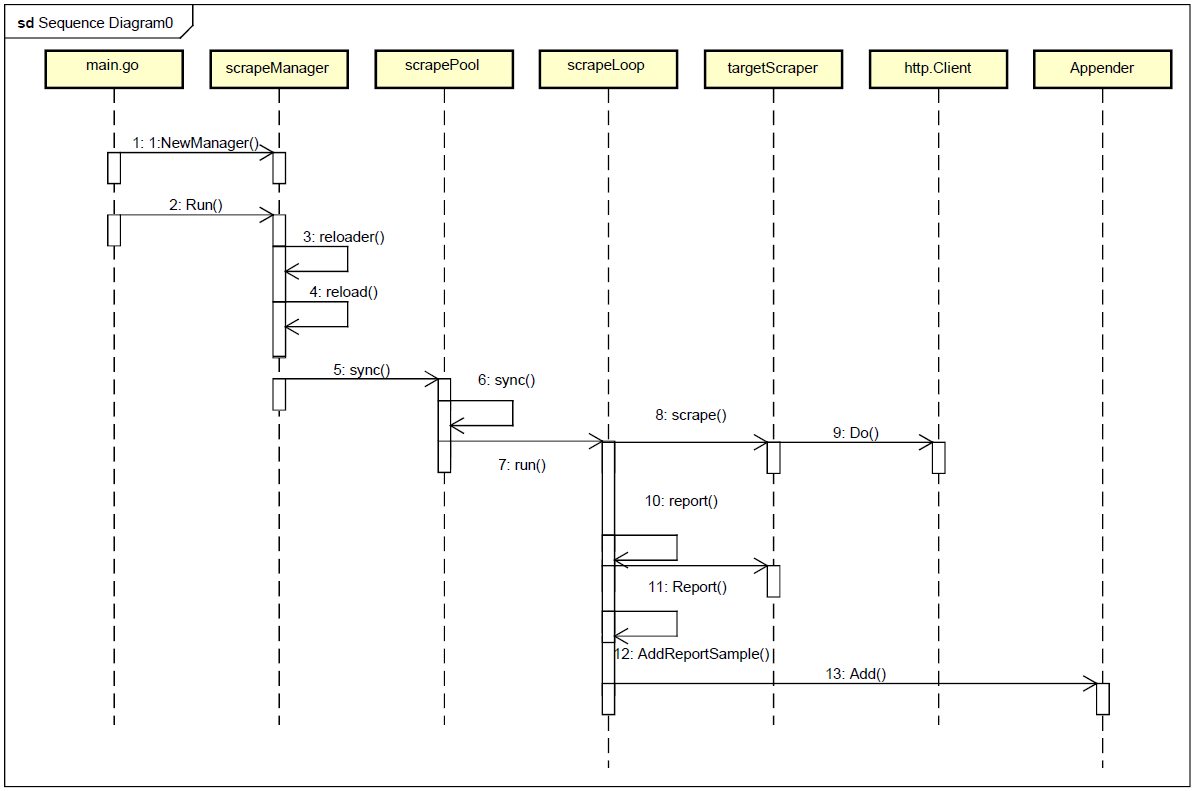


圖 3：Scrape Sequence Diagram

Scrape時序圖(圖 3)分析

1. main.go中使用scape.NewManager建立ScrapeManager，ScapeManager中使用Map結構來儲存管理目標資料採集的scrapePools, 採集配置scrapeConfig，目標集合targetSets；Prometheus 中，將一個獨立的數據來源（target）稱之爲instance。包含相同類型的instance的集合稱之爲job，Manager中ApplyConfig(…)方法可讀取抓取配置，使用job爲key查詢scrapeConfig Map結構，如果job不存在，從ScrapePool中刪除，如果配置已被更改，清理歷史配置，啟動reload將新配置重新加載到ScrapePool。
2. 建立 ScrapeManager之後，main.go 呼叫scrapeManager.Run() 啟動ScrapeManager， Run()方法中先執行reloader() 加載targets，如果targets更新，會觸發重新加載reload()，加載完成後呼叫Sync()將當前運行的抓取器與結果集同步。
3. Manager的reloader()方法用來加載targets，reloader的加載發生為背景執行，不會影響target的更新。
4. Manager的reload() 執行重新加載，爲targets集合中每一個target資料來源生成一個對應的scrape pool來管理其運作，由targetSets中取得所有的Job，若處理Job的對應scarpe pool不存在於Scrape\_pools之中，則讀取Job對應的scrape config配置以建立scrape pool，並將其儲存於Scrape\_pools Map之中，為提高效率，此流程採並行運行，需先用sync.Mutex鎖定，遍歷TargetGroup 將所有scrape pool創建完成後釋放鎖，並等待其他並行工作運行完成。
5. 呼叫Scrape Pool 的Sync()方法，會將當前運行的抓取器與All集合同步，sync.Mutex鎖定後，遍歷所有TargetSet群組中的Target，有效的Target加入All集合，無效的Target加入droppedTargets集合，接著呼叫sync()方法將All集合中Target轉換為實際的抓取器。
6. Scrape Pool 的sync()方法所取得的Target的列表all可能重複，需將重複刪除，並為新目標啟動抓取循環，並為消失的目標停止抓取循環。完成後對每一個不重複的Scrape Loop 呼叫run() 方法執行抓取循環。
7. Scrape Loop 的run() 方法執行抓取循環，依據每個Scrape Loop 定義的時間週期，定期呼叫scrapeAndReport()方法執行抓取。scrapeAndReport()方法會執行抓取，依序觸發執行targetScraper的scrape()方法、http:Clinet的Do()方法、ScrapeLoop的report()方法。然後將結果與報告指標附加記錄到附加器Appender的末端。抓取動作可能使用到一個以上的Appender，Scrape Loop 會盡可能使用較少的 appender。
8. targetScraper的scrape()方法透過HTTP Get請求來抓取數據，先建立HTTP Request請求物件，設定適當請求標頭，透過http:Clinet的Do()方法來發起請求。
9. 執行http:Clinet的Do()方法，傳回抓取資訊。
10. ScrapeLoop的report()方法會呼叫targetScraper的Report()方法抓取的訊息，並針對結果使用addReportSample()方法做處理或錯誤回報。
11. 執行targetScraper的Report()方法，回報關於最後一次抓取的目標數據
12. addReportSample()方法呼叫Appender Add()方法將資料寫入儲存體。
13. 執行Appender Add()方法，storage.Appender 針對存儲體提供資料批次附加的邏輯。必須通過調用 Commit 或 Rollback 來完成，之後不得重用。