Purpose

本文檔描述瞭如何使用 Nagios XI 中的事件處理程序在您監視的主機或服務更改狀態時執行預定義的操作。
事件處理程序用於在特定主機或服務發生狀態更改時自動執行流程。
這很有用，因為當您的環境發生變化時，它可以減少手動工作量。
本文檔還包括一些有關高級配置和腳本的提示。

目標聽眾

本文檔供想要實現自定義事件處理程序的 Nagios XI 管理員使用除了簡單的停止和重新啟動服務之外的腳本。
shell 腳本的基本功能知識和 Nagios 巨集是推薦的。

先決條件

本文檔假設已安裝 Nagios XI 並設置了一些主機和服務。
此外，核心配置管理器 (CCM) 的知識和對一般 Nagios XI 工作流程的理解將
也有幫助。。

什麼是事件處理程序？

事件處理程序是可選的系統命令（腳本或可執行文件），當主機或服務狀態發生變化。
這些命令包括但不限於重啟服務、解析日誌、檢查其他主機或服務狀態，進行數據庫調用等。
可能性幾乎是無限的。
本質上，通過事件處理程序， Nagios XI 能夠運行任何可以從具有使用 Nagios XI 傳遞的巨集的附加能力的命令行

什麼時候在 Nagios XI 中執行事件處理程序？

每當發生狀態更改時都會調用事件處理程序。
這包括 HARD 和 SOFT 狀態類型，以及 OK 、 WARNING 、 CRITICAL 和 UNKNOWN 狀態。
狀態更改發生時採取哪些操作（如果有）的邏輯由 Nagios XI 在狀態更改時調用的腳本或可執行文件執行。
腳本可以通過事件處理程序傳遞給它的巨集來解析這些狀態。

例如，可以創建一個事件處理程序，當且僅當該服務進入硬臨界狀態。
這可以通過 $SERVICESTATE$ （ OK 、 WARNING 、 UNKNOWN 、 CRITICAL ）巨集
和 $SERVICESTATETYPE$ （ HARD 、 SOFT ）巨集傳遞給腳本。
編寫腳本的人有責任確保腳本邏輯正確處理傳遞的巨集。

事件處理程序可以傳遞什麼類型的巨集？

任何標準 Nagios 巨集都可以通過事件處理程序傳遞給腳本或二進製文件。
通常情況下巨集與事件處理程序的服務或主機相關，但您可以將巨集傳遞給引用其他主機和服務的腳本。
作為參考，可以在以下位置找到 Nagios 巨集列表：

https://assets.nagios.com/downloads/nagioscore/docs/nagioscore/4/en/macrolist.html

工作事件處理程序的最低要求

事件處理程序有幾個不同的部分：

- Nagios XI 中的 check 命令本身
- 從 Nagios XI 中的 check 命令調用的事件處理程序腳本

更複雜的事件處理程序還可以將巨集傳遞給事件處理程序腳本並引用遠程腳本。

入門 – 在 Nagios XI 中設置您的第一個事件處理程序

為簡單起見，本文檔將使用 bash 腳本，即使事件可以運行二進製文件、 perl 、 python 、 php 和 bash 腳本等。

第一個事件處理程序將是一個非常簡單的 bash 腳本。

當主機狀態發生變化時，它本質上會將主機信息寫入本地文本文件。

隨著文檔的進展，我們將向該腳本添加功能。

此示例將遵循以下一般步驟：

I. 在XI中為事件處理程序創建一個命令

II. 創建一個虛擬主機進行測試

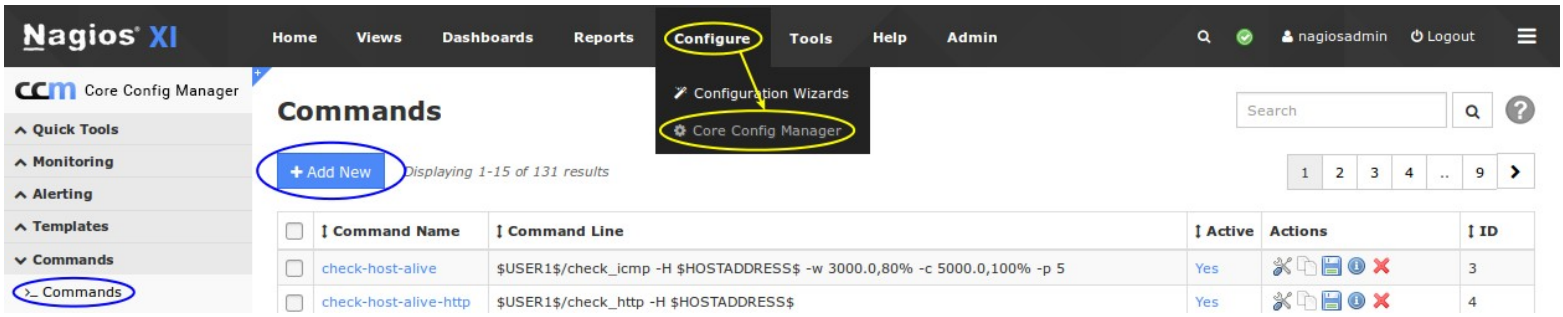
III.在libexec目錄中創建由處理程序運行的腳本

IV. 測試事件處理程序

V.添加高級功能和巨集

I. 在 Nagios XI 中為事件處理程序創建一個命令

在 Nagios XI Web 界面中，導航到 Configure > Core Config Manager > Commands.



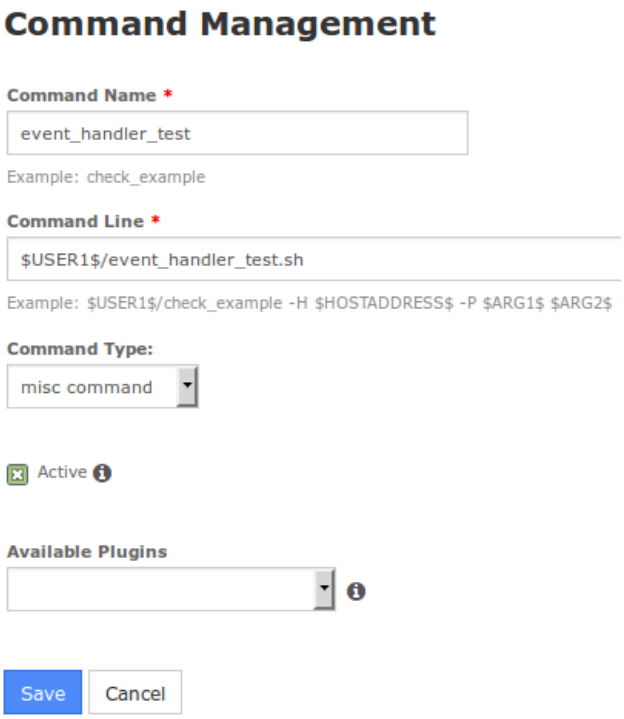
單擊 Add New 按钮，您將需要提供以下詳细信息：

Command Name: `event_handler_test`

Command Line: `$USER1$/event_handler_test.sh`

Command Type: `misc command`

Make sure the Active box is checked.



最終的命令定義應類似於右側的屏幕截圖。

完成後點擊 Save 。

$USER1$ 從 resource.cfg 文件中引用目錄 /usr/local/nagios/libexec 。

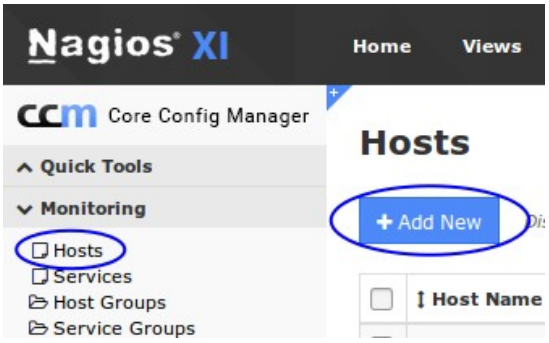
這是 Nagios XI 中插件和腳本的默認路徑。 .

稍後我們將創建一個腳本以供此命令使用，該腳本的名稱將是 event_handler_test.sh 。

II. 創建虛擬主機

在 CCM 中導航到 Monitoring > Hosts.

單擊 Add New 按鈕，您需要提供以下詳細信息：



Common Settings 選項卡

Host Name: `event_handler_test`

Address: `127.0.0.1`

Check command: `check_dummy`

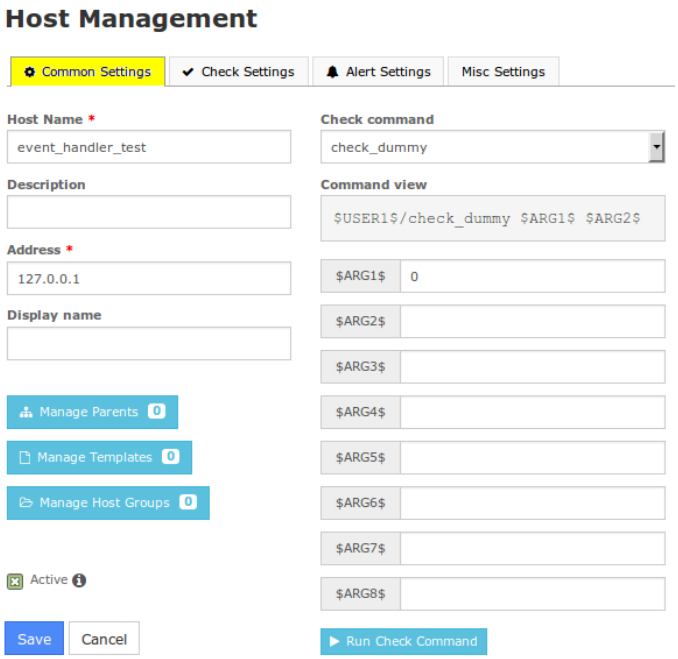
$ARG1$: `0`

這會強制 check_dummy 命令始終返回 0 (UP)

單擊運行 Ceck command 應輸出 OK 。

這意味著您已正確設置命令。

選中 Active 複選框。



Check Settings 選項卡

Check interval: `5`

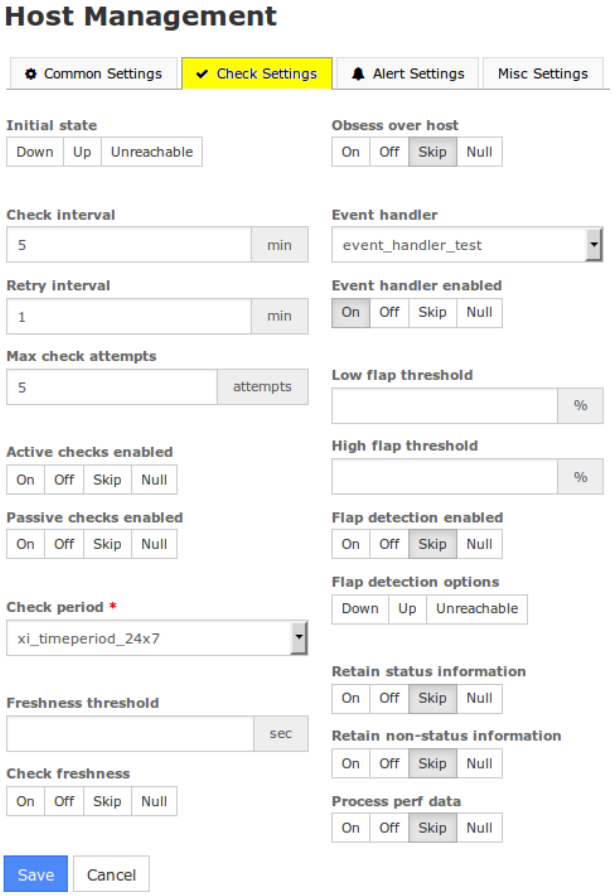
Retry interval: `1`

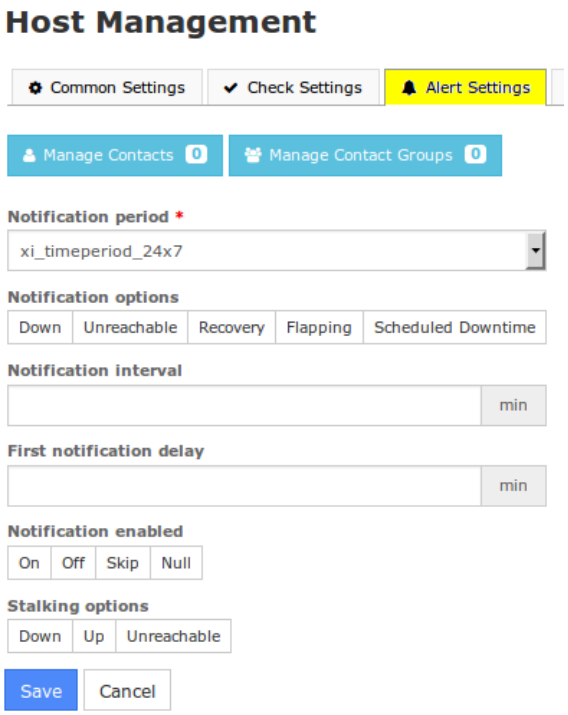
Max check attempts: `5`

Check period: `xi_timeperiod_24x7`

Event handler: `event_handler_test`

Event handler enabled: `On`





Alert Settings 選項卡

Notification period: `xi_timeperiod_24x7`

注意：您可以在主機級別（即 notification_period ）本地定義對象的指令或從主機模板（即 notification_interval 、 notifications_enabled 等）繼承值，如圖所示。

單擊 “ Save” 按鈕以創建主機。

單擊 Apply Configuration 按鈕應用新配置，這樣新 event_handler_test 主機將變為活動狀態。

III. 創建由處理程序運行的腳本

現在 Nagios XI 事件處理程序和宿主對像已經配置完畢，我們需要創建一個由處理程序運行的腳本。
腳本的第一次迭代只會將一些非常基本的信息輸出到 /tmp/ 目錄中的文本文件。
在本文檔的後續部分中，我們將為處理程序的腳本和命令（主要是巨集和腳本邏輯）添加功能和復雜性。
在我們進入有趣的東西之前，讓我們確保基本腳本有效。

在以下所有步驟中，您將被指示創建和更新腳本。

這將使用 vi 文本編輯器完成。使用 vi 編輯器時：

要進行更改，請先按鍵盤上的 i 進入插入模式 進行所需的更改

按鍵盤上的 Esc 退出插入模式

鍵入 :wq 保存更改並退出 vi 編輯器

打開與 Nagios XI 服務器的終端會話並以 root 用戶身份登錄並執行以下命令：

```
cd /usr/local/nagios/libexec
touch event_handler_test.sh
chmod +x event_handler_test.sh
```

現在鍵入以下命令以使用 vi 編輯器編輯文件：

```
vi event_handler_test.sh
```

將以下內容粘貼到文件中：

```
#!/bin/bash
DATE=$(date)
echo "The host has changed state at $DATE" > /tmp/hostinfo.txt
```

保存更改。

IV. 測試事件處理程序

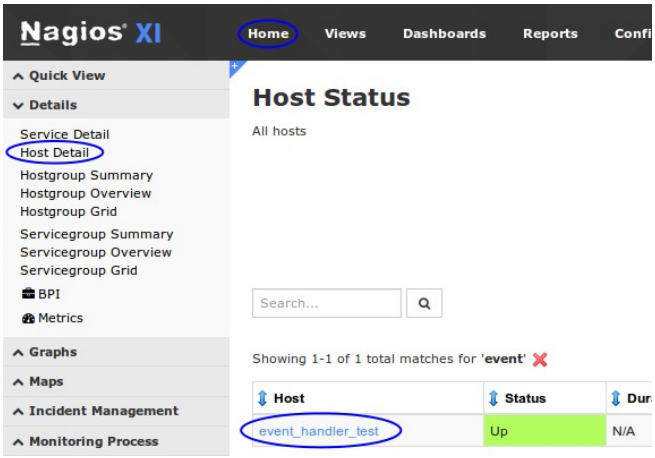
現在一切就緒，我們需要強制虛擬主機進入停機狀態。

通過進入關閉狀態，事件處理程序將被觸發運行。

這將創建文件 /tmp/hostinfo.txt ，因為這是 event_handler_test.sh 腳本將執行的操作。

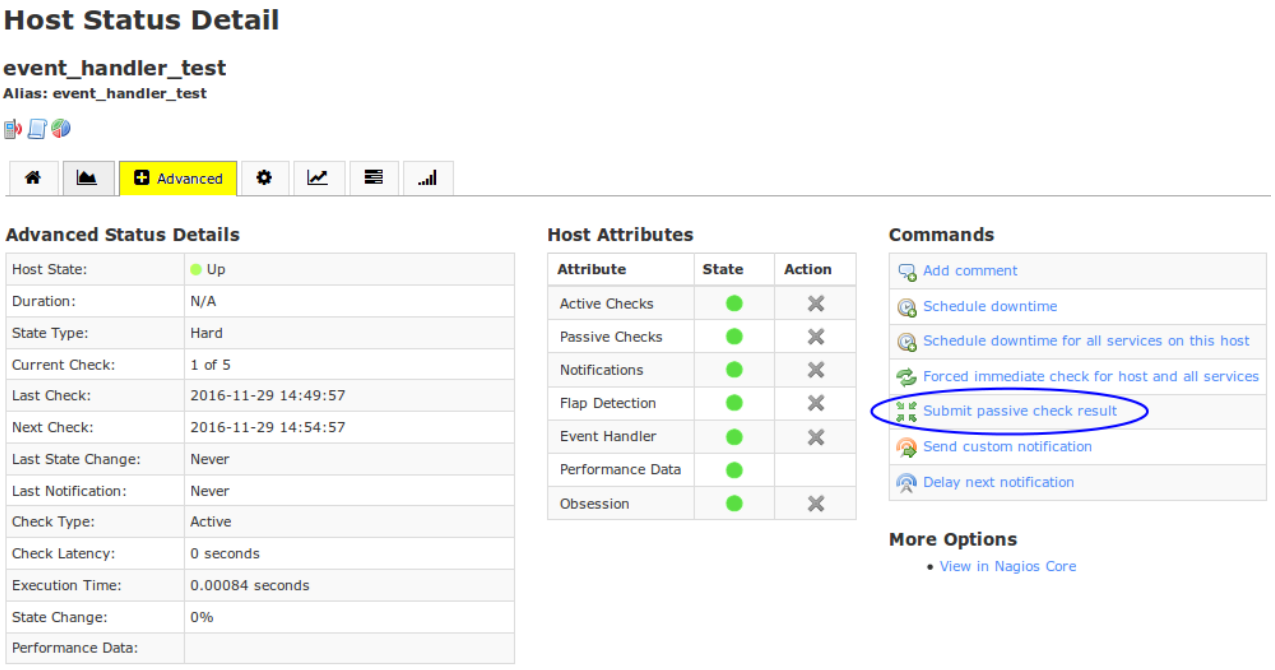
將主機置於停機狀態將通過提交帶有 DOWN 檢查結果的被動檢查來完成。

在 Nagios XI Web 界面中導航到 Home > Host Detail



點擊 host event_handler_test.

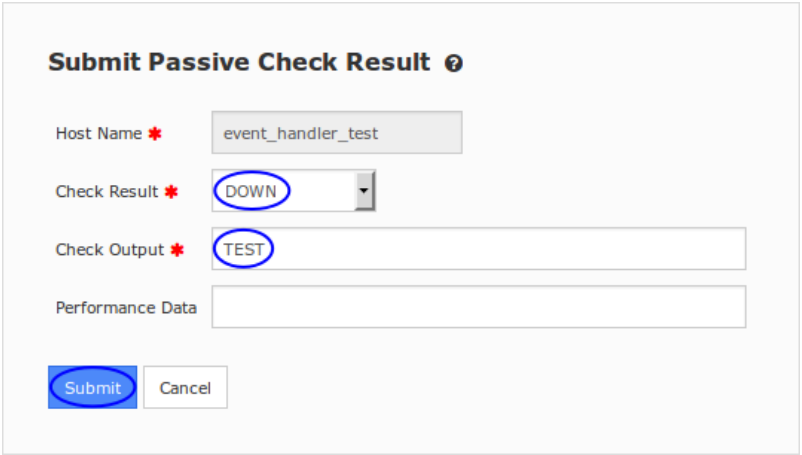
單擊 Advanced 選項卡，然後在 Commands 表下單擊提交 Submit passive check result 。



點擊 Submit Passive Check Result 會出現：

Check Result: DOWN

Check Output: TEST



點擊 Submit.

一兩分鐘後，您應該會看到主機變為停機狀態。

現在讓我們檢查 /tmp/hostinfo.txt 文件是否是通過從 Nagios XI 服務器終端會話，運行以下命令創建的

```
cat /tmp/hostinfo.txt
```

你應該看到類似的輸出：

```
The host has changed state at Tue Nov 29 15:10:01 AEDT 2016
```

恭喜，您已經創建了第一個事件處理程序。
這證明了當主機狀態改變時 Nagios 執行了事件處理程序。

這些第一部分實際上只是皮毛，但在您嘗試實現一些高級功能
（例如傳遞巨集並在腳本邏輯中使用它們來執行諸如重新啟動服務或 執行其他自定義腳本。
您甚至可以使用事件處理程序通過命令管道將命令傳回 Nagios 。
如果您不確定前面的步驟，請花點時間重新閱讀它們，因為從現在開始事情會變得更加複雜。

繼續閱讀文檔的下一部分以了解事件處理程序的更高級功能。

V. 高級功能和巨集

現在您已經擁有一個功能齊全的事件處理程序設置，您可能想用它做一些事情。
我們在前面的步驟中創建的腳本實際上並沒有做太多事情。
下一步將展示巨集如何工作，因為它們是充分發揮事件處理程序潛力的關鍵。
有許多巨集可以傳遞給 Nagios 中的事件處理程序腳本，儘管它們是特定於對象的，
因此一些巨集僅在從服務對像傳遞時才起作用，其他巨集從宿主對像傳遞，而其他巨集可以從兩者傳遞。

需要編輯事件處理程序命令定義以定義將傳遞給事件處理程序腳本的巨集。
在下一步中，我們將向 event_handler_test 命令添加一些巨集，
然後向我們的腳本添加一些邏輯以將這些巨集打印到 /tmp/hostinfo.txt 文件中。
Nagios 核心文檔網站上的巨集列表包含所有可用的 Nagios 巨集以及哪些對象可以傳遞它們的相當全面的列表：

https://assets.nagios.com/downloads/nagioscore/docs/nagioscore/4/en/macrolist.html

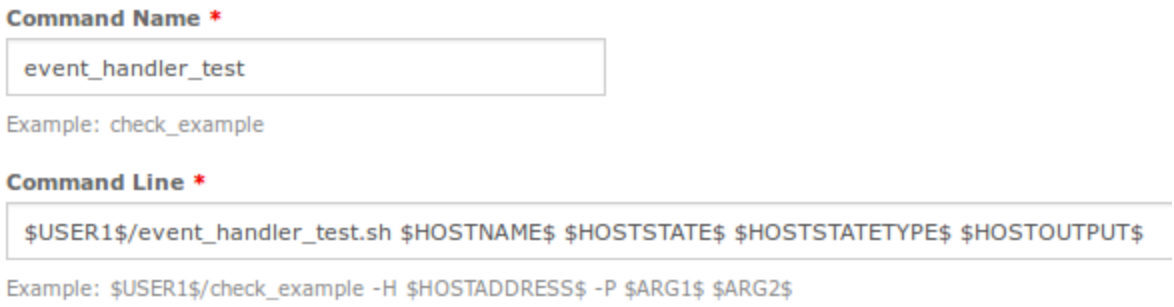
添加 Macros

需要注意的一件事是，您只能傳遞對像類型支持的巨集，並且由於我們正在使用主機檢查，
因此我們僅限於主機對象宏巨集。
您很可能希望包含在大多數（如果不是所有）（主機）事件腳本中的最重要的巨集是：

- `$HOSTNAME$`
- `$HOSTSTATE$`
- `$HOSTSTATETYPE$`
- `$HOSTOUTPUT$`

在 Nagios XI 中，導航到 Configure > Core Config Manager > Commands
並編輯 event_handler_test 命令。

將上述四個巨集按順序追加到命令行中，用空格隔開



完成後，保存並套用配置。

現在腳本需要更新。 返回到 Nagios XI 服務器上的終端會話並鍵入以下內容
使用 vi 編輯器編輯文件的命令：

```
vi event_handler_test.sh
```

Paste the fo 將以下內容粘貼到文件中：

```
#!/bin/bash
HOSTNAME=$1
HOSTSTATE=$2
HOSTSTATETYPE=$3
HOSTOUTPUT=$4
DATE=$(date)
echo "The host $HOSTNAME has changed to a $HOSTSTATETYPE $HOSTSTATE state at
$DATE with the error: $HOSTOUTPUT" > /tmp/hostinfo.txt
```

注意：最後兩行是一長行，它們只是在本文檔中被分成兩行。

保存更改。

Bash 將接收巨集作為參數，按順序枚舉它們，給它們命名為 $1 、$2 、$3 和 $4 。
為了更容易理解它們在腳本中的工作方式，我們將使用原始巨集名稱將它們分配給變量。

從更改中可以看出，對於腳本，變量最初定義為單詞 (HOSTNAME)，
但它被帶有 $ 符號 ($HOSTNAME) 的 echo 命令引用。
您可以從 echo 行中看到我們如何使用變量將變量中的數據輸出到 /tmp/hostinfo.txt 文件。
通過提交被動檢查結果（按照 IV. 測試事件處理程序 中的步驟）再次強制主機進入停機狀態。
提交被動檢查後，等待一兩分鐘，然後檢查 /tmp/hostinfo.txt 文件的內容：

```
cat /tmp/hostinfo.txt
```

再一次，您應該看到類似的輸出：

```
The host event_handler_test has changed to a HARD DOWN state at Tue Nov 29
16:48:57 AEDT 2016 with the error: TEST
```

如果您的輸出類似於上面的行，那麼恭喜！ 您可以看到巨集中的數據以粗體顯示以突出顯示它們已被使用。
您現在有一個帶有巨集的工作事件處理程序。
但是，我們還沒有完成，因為這個文本文件仍然沒有做太多事情。
我們需要向腳本添加一些邏輯來檢查巨集中的某些字符串並執行一兩個任務。

添加邏輯和執行任務

大多數事件處理程序需要的第一部分邏輯是分離出主機或服務對象的不同狀態。
此外，檢查 HARD 或 SOFT 狀態通常是一個好主意，
因為大多數處理程序僅在對象處於 HARD 問題狀態時才需要。
大多數情況下，您的事件處理程序會觸發一個額外的腳本，該腳本實際執行您關心的任務，
有時會向該腳本傳遞額外的參數。

下一頁是執行以下操作的腳本的更新版本：

- 將接收到的巨集值分配給變量
- 將當前日期分配給變量
- 使用 if 運算符來確定這是否是 SOFT 狀態，如果是，則腳本退出，
  因為當它處於 SOFT 狀態時我們不想做任何事情
- 使用 case 語句執行不同的命令，具體取決於 $HOSTSTATE 是 UP 、 DOWN 還是
- 在每個不同的 case 語句中，您可以看到它正在寫入磁盤上的不同文件
  （ UP.txt 、 DOWN.txt 或 UNREACHABLE.txt ）。
  雖然此功能是基本功能，但它展示了根據從 Nagios 收到的巨集的值執行不同操作的能力。

```
#!/bin/bash
HOSTNAME=$1
HOSTSTATE=$2
HOSTSTATETYPE=$3
HOSTOUTPUT=$4
DATE=$(date)
if [ $HOSTSTATETYPE == 'SOFT' ]
then
   exit 0
fi
case "$HOSTSTATE" in
   UP)
   echo "The host $HOSTNAME changed to a $HOSTSTATE state @ $DATE" > /tmp/UP.txt
   ;;
   DOWN)
   echo "The host $HOSTNAME changed to a $HOSTSTATE state @ $DATE" > /tmp/DOWN.txt
   ;;
   UNREACHABLE)
   echo "The host $HOSTNAME changed to a $HOSTSTATE state @ $DATE" > /tmp/UNREACHABLE.txt
   ;;
esac
```

進行這些更改後，嘗試提交一些更被動的檢查結果，以便您可以看到它如何寫入磁盤上的不同文件。

事件處理程序腳本用例示例

大多數情況下，事件處理程序只需要為某種狀態運行一個特定的命令。
一個很好的例子是，如果服務處於臨界狀態，則在 Windows 或 Linux 機器上重新啟動該服務。
完整的工作示例可以在以下文檔中找到：

- Restarting Windows Services With NCPA
- Restarting Linux Services With NCPA
- Restarting Windows Services With NRPE
- Restarting Linux Services With NRPE

傳遞巨集時運行外部腳本

另一個最常見的用例是使用事件處理程序將巨集傳遞給另一個外部腳本。
這可以包括重啟某些服務的腳本、其他 Nagios 插件腳本（包括 check_nrpe 或 check_by_ssh ），
甚至是與外部 Ticketing System 通信的腳本。
事件處理程序僅受您的想像力的限制。

下面的事件處理程序腳本將假設我們要將巨集傳遞給外部腳本，
該腳本將為票務系統準備警報到 Ticketing System 理解的語法中。
以下腳本中的邏輯只會將問題傳遞到關鍵的票務系統。
參數中包含一些更晦澀的巨集，例如 $SERVICEPROBLEMID$ ，它是 Nagios 中每個問題的唯一編號。
這就是我們的 Ticketing System 跟踪 Nagios 報告的問題的方式。
與前面的示例一樣，此示例將是一個服務事件處理程序。
如果要為主機問題打開票證，則必須創建另一個處理程序，或者擴展以下腳本的邏輯。

通過電子郵件發送到 Ticketing System ：
許多 Nagios XI 管理員已經維護了一個單獨的 Ticketing System ，用於管理技術人員的停電。
將 Nagios XI 警報與這些系統集成通常是通過特製的電子郵件完成的。
這可以通過事件處理程序腳本來完成，該腳本使用 CLI 電子郵件實用程序 sendmail 發送自定義格式的電子郵件。

核心通知處理程序。

以下事件處理程序和腳本只是一個模型，您絕對需要編輯電子郵件的格式以與您的票務系統兼容。

Ticketing System Service 事件處理程序命令行：

```
$USER1$/event_handler_service_ticket.sh "$HOSTNAME$" "$SERVICEDESC$"
$SERVICESTATE$ $SERVICESTATETYPE$ $SERVICEPROBLEMID$
```

The script `event_handler_ticket.sh` is as follows:

```
#!/bin/bash
DATE=$(date)
HOSTNAME="$1"
SERVICEDESC="$2"
SERVICESTATE=$3
SERVICESTATETYPE=$4
# SERVICEPROBLEMID and HOSTPROBLEMID are unique and are by far the best way to
track issues detected by Nagios
SERVICEPROBLEMID=$5
# Set your ticketing system's email below
EMAIL="email@domain.tld"
# The message below is a mock up.  You will need to edit the format to one
acceptable to your ticketing system
if [[ "$SERVICESTATETYPE" == "HARD"  && "$SERVICESTATE" == "CRITICAL" ]]; then
        /usr/bin/printf "%b" "***** Nagios Monitor XI Alert *****\n\nHost:
$HOSTNAME\nState:  $SERVICESTATE\n\nDate/Time: $DATE\n" | /bin/mail -s "OPEN
TICKET: $SERVICEPROBLEMID - $SERVICEDESC on $HOSTNAME is $SERVICESTATE " $EMAIL
        exit 0
else
        exit 0
fi
```

此腳本僅在相關服務處於 HARD 、 DOWN 狀態時提交票證。
printf 命令從 "%b" 到管道的部分是電子郵件的正文，而從 "/bin/mail -s" 到 $EMAIL 宏的字符串是主題。
大多數票務系統只會查看傳入電子郵件的主題來打開票證，儘管有些系統也可以解析消息的正文。

其他註釋、選項和其他資源

Nagios XI 中的事件處理程序系統健壯、可擴展且高度靈活。
Nagios XI 中有一些更晦澀且經常被忽視的選項，可用於擴展或更改事件處理程序的行為。

Is Volatile 檢查屬性

此選項位於 CCM 中任何服務管理頁面的 “ Check Settings” 選項卡底部。
啟用此選項將強制 Nagios XI 將對相應對象的每個檢查都視為狀態更改。
這基本上會在每次檢查對象時運行事件處理程序，無論是計劃的主動檢查，
還是接收到的被動檢查，並且它與狀態無關（無論對象狀態如何，事件處理程序都將始終運行）。
為具有事件處理程序的對象啟用此選項後， Nagios XI 實際上可以用作日常維護或更具體任務的調度程序。
更多信息可以在以下鏈接中找到：
https://assets.nagios.com/downloads/nagioscore/docs/nagioscore/4/en/volatileservices.html

Free variables

有時，普通對象巨集不足以滿足您的腳本和事件處理程序需求。
Nagios XI 包括指定要用作巨集的對像或模板特定變量的選項。它們位於對象的 “ Misc Settings“ 選項卡上。
Free variables 必須以下劃線為前綴，以避免與普通變量 / 巨集的命名衝突問題。
您可以通過將這些變量視為傳遞給事件處理程序的巨集來使用這些變量，
方法是將它們作為 $_YOURVARIABLE$ 在對像上定義後引用。
有關更多信息，請參閱以下鏈接：
Using The Nagios XI Core Config Manager For Host Management
Using The Core Config Manager For Service Management
https://assets.nagios.com/downloads/nagioscore/docs/nagioscore/4/en/customobjectvars.html

On-Demand Macros (Cross-Object)

複雜的事件處理程序腳本偶爾需要訪問來自其他對象的巨集。

用例包括故障轉移場景、彈性集群、負載平衡等。通常巨集只引用作為相關對象值的變量。

有時，開發人員會想要檢查其他對象巨集的值。

例如，在故障轉移場景中，您的事件處理程序可能想要檢查與觸發事件的主機不同的主機上特定服務的狀態。

如果您想在事件處理程序中引用另一個主機或服務的值，您可以使用所謂的“On-Demand Macros”。

On-Demand Macros 看起很普通，不同之處在於它們包含主機或服務的標識符，它們應該從中獲取它們的值。

以下是 On-Demand Macros 的基本格式：

```
$HOSTMACRONAME:host_name$
$SERVICEMACRONAME:host_name:service_description$
```

請注意，巨集名稱與主機或服務標識符之間用冒號 (:) 分隔。

對於 On-Demand Macros 巨集，服務標識符由主機名和服務描述組成 - 它們也由冒號 (:) 分隔。

按需主機和服務巨集的示例如下：

```
HOSTDOWNTIME:myhost$
$SERVICESTATEID:novellserver:DS Database$
$SERVICESTATEID::CPU Load$
```

可以在以下鏈接中找到有關 On-Demand Macros 的更多信息：

https://assets.nagios.com/downloads/nagioscore/docs/nagioscore/4/en/macros.html

BASH 腳本教程

儘管可以使用任何編程 / 腳本語言，但 bash 腳本是最容易為其創建的一些腳本。
簡單的任務，不需要重新編譯，但如果需要，仍然可以完成複雜的任務。
在本文檔中，所有使用的示例都是在 bash 中完成的。
如果您不熟悉腳本編寫，那麼 bash 是一個很好的起點，
尤其是因為本文檔包含許多腳本模板，以便於在 Nagios XI 中與事件處理程序一起使用。
如果您想了解有關 bash 腳本的更多信息，可以充分利用您最喜歡的搜索引擎或查看以下鏈接：

http://www.tldp.org/LDP/Bash-Beginners-Guide/html/
https://help.ubuntu.com/community/Beginners/BashScripting

整理起來

這完成了有關如何使用 Nagios XI 中的事件處理程序
在您監視的主機或服務更改狀態時採取預定義操作的文檔。
如果您有其他問題或其他與支持相關的問題，請訪問我們的 Nagios 支持論壇：

https://support.nagios.com/forum

Nagios 支持知識庫也是一個很好的支持資源：

https://support.nagios.com/kb