Installing and Using the Vim Text Editor on a Cloud Server

Authored by: ***ASPHostServer Administrator*** *[asphostserver@gmail.com]*
Saved From: *http://faq.asphosthelpdesk.com/article.php?id=203*

Introduction

One of the most powerful text editors accessible from the command line is the vim editor. Built on the foundation of "vi", an editor dating back to 1976, vim adds additional functionality and power, while maintaining the editing style of its predecessor.

Installation

Due to vim's wide-spread use on the Linux command line, it is available in almost every distribution's default repositories.

On Ubuntu and Debian, use apt-get to install:

```
sudo apt-get install vim
```

On Fedora and CentOS, install with yum:

```
sudo yum install vim
```

On Arch Linux, vim can be install through pacman:

```
sudo pacman -S vim
```

Vim should now be installed correctly.

Opening Vim

By default, when vim is called without any arguments, it opens with a blank document. Depending on your distribution, there may be an introduction and licensing message. This will disappear when you begin using the editor.

However, we are not going to open vim with a blank document. We will create a sample document called "newfile" to showcase vim's features.

```
echo "This is a new file.
Here is the second line
And here is the third line.
Some more text is on this line.  We will edit this file in vim.
It will be great fun." >> newfile
```

Now, we will open the file we just created with vim.

```
vim newfile
```

Modal Editing

The main difference between vim and most other editors is that vim is a "modal" editor. Most other editors have only one mode. In these editors, special editing functions, like copying text, are performed by holding one or more modifier keys and then hitting a regular key. Vim uses distinct modes to separate these functions from the task of normal text input.

Normal Mode

Used for editing operations. Copying, pasting, moving, deleting, and changing text is accomplished from within this mode.

In vim, editing functions are performed by putting vim into "normal" mode. Normal mode is the mode that vim is in upon opening the program. This mode is used to navigate a text document quickly and to perform editing. It is not used for entering text.

[esc] - Enter normal mode by pressing the "escape" key.

Insert Mode

Used for inserting new text in the document. It is possible to enter insert mode in a number of ways.

To enter text, vim must transition into "insert" mode. Insert mode is analogous to the typing interface of most other text entry programs. What you type appears on the screen in the document. All normal keys will produce the corresponding character at the current cursor position.

i - Enters insert mode at the current cursor position.
a - Enters insert mode after the current position.
I - Enters insert mode at the beginning of the current line.
A - Enters insert mode at the end of the current line.

Visual Mode

Used for visual selection. Many commands available in normal mode can be applied to a specific highlighted portion of the text.

A third mode that vim uses is "visual" mode. This is used for visual selection and manipulation of text. Areas of text are highlighted as a target of subsequent editing or formatting commands.

v - Enters regular visual mode. Selections are made by moving the cursor up, down, left, and right.
V - Enters visual line mode. Entire lines are selected, from the first to the last character, by moving up and down.
[ctrl]-v - Enters visual block mode. A box is used for selection that can be expanded and contracted. Portions of multiple lines can be selected with this method.

Command Mode

Used for issuing vim commands. Enter this mode with the colon key.

An additional mode that is used for complex edits, changing settings, and controlling vim itself is "command" mode. This mode is used for saving documents, quitting the program, performing complex searches and many other things.

: - Enters command mode.

Navigation

Basic Navigation

It is always possible to navigate text with the arrow keys, but vim provides faster ways of moving around a document. In normal mode, you can use the h,j,k, and l keys to move left, down, up, and right, respectively.

h - Move left.
j - Move down.
k - Move up.
l - Move right.
These direction keys may seem confusing and counterintuitive at first, but they were chosen for a reason. They are in the home-row of a typical keyboard. This means that a user's hand moves from the resting position significantly less than with the traditional arrow keys.

Advanced Navigation

There are also other navigation shortcuts. Here are a few of the most useful ones:

gg - Move to the top of the document.
G - Move to the bottom of the document. Preface with a number to go to that line number.
w - Move to the next word. Preface with a number to move that many words.
b - Move back one word. Preface with a number to move back that many words.
e - Move to the end of the word. Preface with a number to move that many words.
- Move to the beginning of the line.
$ - Move to the end of the line.

Editing

Editing text in vim is accomplished by issuing commands in normal mode.

It is important to realize that editing commands in vim are very powerful when combined with motion commands. Anything from the navigation section can be used as a direction. For instance, you can perform an editing command on a word following it with "w".

Here are a few of the different actions you can choose from:

Removing text

---

x - Delete the character under the cursor position.
d - Deletes in the direction you specify after issuing the command. For example, "dl" deletes a character to the right.
dd - Delete a line.
D - Delete from the current position to the end of the line.

Changing Text

---

r - Replace the character under the cursor with a different character. Give the character you would like to substitute after issuing this command.
c - Change the text in the direction that follows. For example, "cw" changes the current word. After issuing this command, vim will be placed in insert mode so that you can supply replacement text.
C - Change the text to the end of the line. This will place vim in insert mode.

Copying and Pasting

---

y - Copy (or "yank") in the direction that follows.
yy - Copy the entire line.
Y - Copy until the end of the line.
p - Paste the last line copied (or deleted!) below the current line.
P - Paste the last line copied (or deleted!) above the current line.

Miscellaneous Editing

---

u - Undo the last action.
<ctrl>-r - Redo the last action.
J - Join the line below to the current line.

Managing Documents

Vim manages documents mainly through its command mode. Commands are issued by typing a ":" before each command.

:q - Quit vim. If changes have not been saved, this will fail.
:q! - Quit vim and discard any unsaved changes.
:w - Save changes. Add a space and a filename if you would like to save to a different location or if this is your first time specifying a save location.
:e - Edit the file that follows.
:bn - Edit the next file that vim has open.
:bp - Edit the previous file that vim has open.

Putting It All Together

As mentioned earlier, vim's power comes from its ability to chain different commands together. The easiest way to grasp this concept is to consider vim commands a language. Keys in normal mode can represent adjectives, verbs, and objects.

For instance, to yank (copy) 4 words, simply translate that phrase into commands that vim recognizes. In this case, it would be:

```
y4w
```

To delete from the current line until the end of the file, type:

```
dG
```

It's also helpful to realize some of the conventions that vim uses for its editing commands. For instance, a doubled letter is usually applied to the entire line. We can copy an entire line with:

```
yy
```

On the other hand, a capitalized version of an editing command often targets from the current cursor position to the end of the line. For instance, to change the text from here until the end of the line, you would use:

```
C
```

To do the same motion on more than one line, just add a numbered prefix to the command. This changes the text from the current position to the end of the following line:

```
2C
```

There are many other powerful techniques to use with vim that are outside of the scope of this guide. If you would like to learn more, a good start is with vim's built in tutorial interface. It will not cover advanced topics, but it will give you a good idea of how to work with vim to do basic editing. You can start it with this:

```
vimtutor
```