

# STATS 507

# Data Analysis in Python

Lecture 16: UNIX/Linux command line

# UNIX/Linux: a (very) brief history

1960s: Multics (Bell Labs, MIT, GE), a time-sharing operating system

1970s: UNIX developed at Bell Labs

1980s: the UNIX wars [https://en.wikipedia.org/wiki/Unix\\_wars](https://en.wikipedia.org/wiki/Unix_wars)

1990s: GNU/Linux emerges

2000s: MacOS developed based on UNIX

Bell labs film about UNIX from 1982:

<http://techchannel.att.com/play-video.cfm/2012/2/22/AT&T-Archives-The-UNIX-System>

# The Unix philosophy: do one thing well

1. Write programs that do one thing and do it well.
2. Write programs to work together.
3. Write programs to handle text streams, because that is a universal interface.

# The Unix philosophy: do one thing well

1. Write programs that do one thing and do it well.
2. Write programs to work together.
3. Write programs to handle text streams, because that is a universal interface.

These three design principles, articulated in the concise form above long after Unix was written, go a long way toward explaining how to approach the command line. For nearly any task you wish to accomplish, there almost certainly exists a way to do it (reasonably) easily by stringing together several different programs. **More information:** [https://en.wikipedia.org/wiki/Unix\\_philosophy](https://en.wikipedia.org/wiki/Unix_philosophy)

# Basic concepts

**Shell** : the program through which you interact with the computer.

provides the command line and facilitates typing commands and reading outputs.

Popular shells: bash (Bourne Again Shell), csh (C Shell), ksh (Korn Shell)

**Redirect** : take the output of one program and make it the input of another.

we'll see some simple examples in a few slides

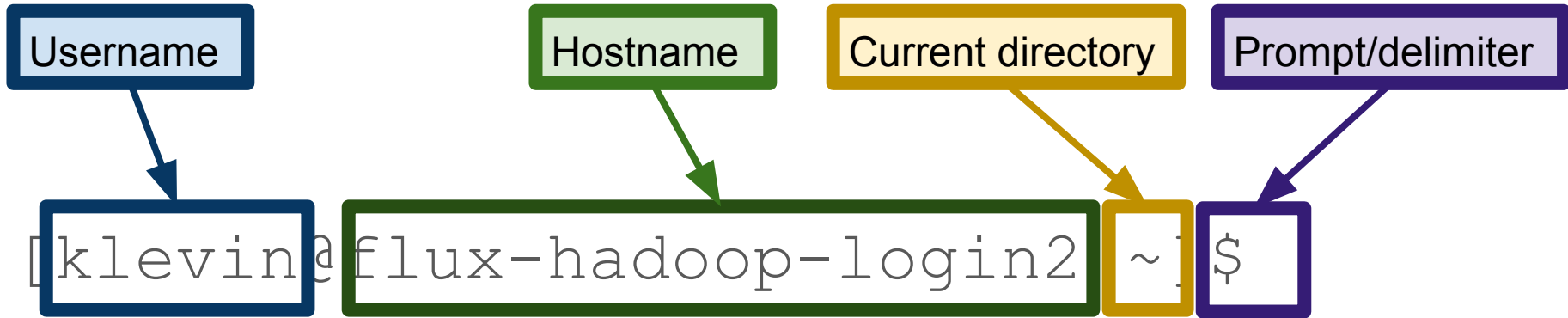


**stdin, stdout, stderr** : three special “file handles”

for reading inputs from the shell (stdin)

and writing output to the shell (stderr for error messages, stdout other information).

# Parts of the command line prompt



**Note:** details of this will vary from one computer to the next (and it can be customized by the user), but this is the default on the Fladoop cluster. For information on customizing the command line prompt, see <https://linuxconfig.org/bash-prompt-basics>

# Connecting to other machines: `ssh`

`ssh` (**S**ecure **S**hell) network protocol allows secure communication machines  
Allows remote access to resources on, e.g., a server or compute cluster

**UNIX/Linux/macOS:** open a terminal, type “`ssh user@machine`”, and you’re off!

**Windows:** `ssh` does not come standard.

PuTTY: <https://en.wikipedia.org/wiki/PuTTY>

Cygwin: <https://en.wikipedia.org/wiki/Cygwin>

# Typical `ssh` session

Secure shell (`ssh`) login to Cavium, from the command line on my Mac (term)

```
keith@Steinhaus:~$ ssh klevin@cavium-thunderx.arc-ts.umich.edu
```

```
Passcode or option (1-2): 2147252
```

```
Success. Logging you in...
```

```
Last login: Thu Oct 17 14:25:01 2019 from 35.1.209.142
```

```
*          Advanced Research Computing – Technology Services  
          University of Michigan  
          hpc-support@umich.edu
```

I cropped a few security-related things out of here.

```
The folders under /scratch are intended for data that is in active use only. Please do not store data there for longer than 60 days. For usage information, policies, and updates, please see: http://arc-ts.umich.edu
```

```
The maintenance window for this cluster is 7am-9am daily. If no jobs are running we may stop services during this time. For updates please follow us on twitter at https://twitter.com/arcts\_um
```

And now I have a command line prompt on the Cavium cluster!

```
[klevin@cavium-thunderx-login01 ~]$ █
```



# Typical `ssh` session

Secure shell (`ssh`) login to Cavium, from the command line on my Mac (term)

```
keith@Steinhaus:~$ ssh klevin@cavium-thunderx.arc-ts.umich.edu
```

```
Passcode or option (1-2): 2147252
```

```
Success
```

```
Last
```

```
*
```

```
The
```

```
use
```

```
For
```

```
ht
```

```
The
```

```
are
```

```
ple
```

If you're using a **Mac or UNIX/Linux** machine, you can pretty much copy what I just did. On Mac, use the app Terminal. On UNIX/Linux systems, you should be able to pull up a terminal using a shortcut like `ctrl+alt+t`, depending on what distribution of UNIX/Linux you're using.

On **Windows**, you can use cygwin to run a command line on your own machine, or use PuTTY to open an `ssh` connection to another machine like I did in this slide.

If you have trouble with any of this, please post to the discussion board and come to office hours to get assistance promptly so that you can do the homework.

lated

And now I have a command line prompt on the Cavium cluster!

```
[klevin@cavium-thunderx-login01 ~]$
```

# Basic commands for navigating

`pwd` : “print/present working directory”. Print the directory that you are currently in.

`ls` : list the contents of the current directory.

**Try this.** Type `pwd` or `ls` in your shell (either in terminal/cygwin or on Fladoop).

`cd dirname` : change the working directory to `dirname`.

Some special directory symbols:

`~` : your home directory. `cd ~` will take you back to your home.

`.` : the current directory. `cd .` will take you to where you are right now.

`..` : the directory above the current directory.

If you're in `/home/klevin/stats`, then `cd ..` will take you to `/home/klevin`.

# Example: pwd, ls and cd

```
keith@Steinhaus:~$ ssh -X klevin@cavium-thunderx.arc-ts.umich.edu
Password:

[...]

[klevin@cavium-thunderx-login01 ~]$ pwd
/home/klevin
[klevin@cavium-thunderx-login01 ~]$ ls
Myfile.txt  stats507f19
[klevin@cavium-thunderx-login01 ~]$ cd stats507f19/
[klevin@cavium-thunderx-login01 stats507f19]$ pwd
/home/klevin/stats507f19
[klevin@cavium-thunderx-login01 stats507f19]$ ls .
hw1.tex  hw2.tex  hw3.tex
[klevin@cavium-thunderx-login01 stats507f19]$ ls ..
myfile.txt  stats507f19
[klevin@cavium-thunderx-login01 stats507f19]$ ls ~
myfile.txt  stats507f19
```

# Getting help: man pages

When in doubt, the shell has built-in documentation, and it tends to be good!

```
man cmdname : brings up documentation about the command cmdname
```

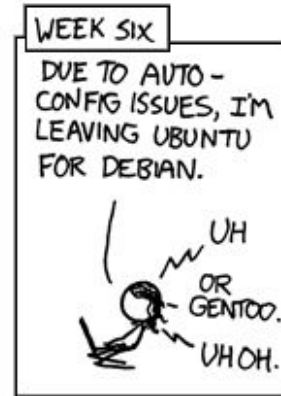
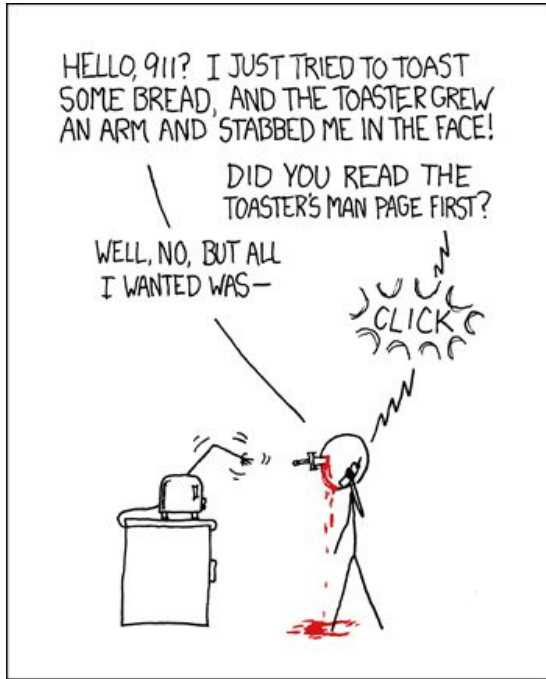
This help page is called a man (short for manual) page. These have a reputation for being terse, but once you get used to reading them, they are extremely useful!

Some shells also have a command `apropos`:

```
apropos topic : lists all commands that might be relevant to topic.
```

**Let's read some of the `ls` man page and see if we can make sense of it.**

# Relevant xkcds



PARENTS: TALK TO YOUR KIDS ABOUT LINUX.. BEFORE SOMEBODY ELSE DOES.

# Basic commands: actually doing things

In the next few slides, we'll look at some commands that actually let you do things like creating files and directories, reading files, and moving them around.

Follow along with the examples in your terminal, if you like (highly recommended).

# Basic commands: echo

`echo string`: prints string to the shell.

```
keith@Steinhaus:~$ echo "hello world."
hello world.
keith@Steinhaus:~$ echo "hello world!"
-bash: !": event not found
keith@Steinhaus:~$ echo "hello world\!"
hello world\!
keith@Steinhaus:~$ echo 'hello world!'
hello world!
keith@Steinhaus:~$ echo "hello\tworld."
hello\tworld.
keith@Steinhaus:~$ echo -e "hello\tworld."
hello    world.
```

The shell tries to interpret the exclamation point as referencing a previous command rather than as text. Escaping doesn't do the trick here. Instead, use single-quotes to tell the shell not to try and process the string.

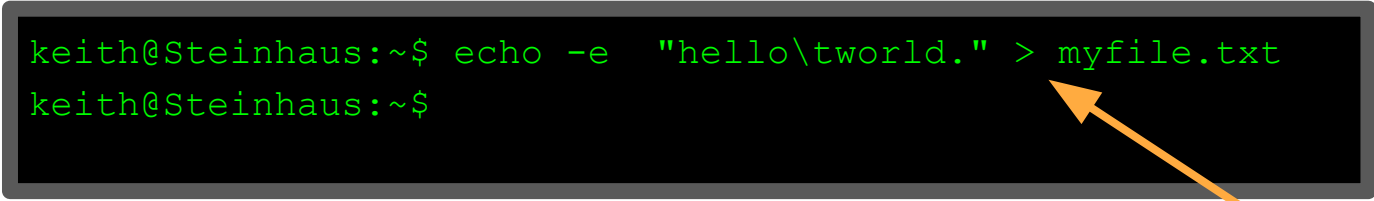
To print special characters (tabs, newlines, etc), use the flag `-e`, without which `echo` just prints what it's given.

**Note:** different shells will have slightly different behavior here, due to differences in parsers.

# Aside: redirections using >

What if I want to send output someplace other than the shell?

```
keith@Steinhaus:~$ echo -e "hello\tworld." > myfile.txt
keith@Steinhaus:~$
```



**Note:** the other redirect, <, has a somewhat similar function, but is beyond our purposes here (stay tuned for command-line workshop at end of semester, perhaps?)

Redirect tells the shell to send the output of the program on the “greater than” side to the file on the “lesser than” side. **This creates the file on the RHS, an overwrites the old file, if it already exists!**



# Basic commands: `cat`

`cat filename` : prints the contents of the file `filename`.

```
keith@Steinhaus:~$ cat myfile.txt
hello    world
keith@Steinhaus:~$
```

So `cat` is like `echo` but it takes a filename as argument instead of a string.

# Basic commands: head

`head filename` : prints the first 10 lines of filename.

`head -n X filename` : prints the first X lines of filename.

```
keith@Steinhaus:~$ head ~/Teaching/Homeworks/HW1/homework1.tex
\documentclass[11pt]{article}

\usepackage{enumerate}
\usepackage{amsmath}
\usepackage{amsfonts}
\usepackage{hyperref}

\oddsidemargin 0mm
\evensidemargin 5mm
\topmargin -20mm
keith@Steinhaus:~$
```

# Basic commands: `more/less`

`more` and `less` are two (very similar) programs for reading ASCII files.

```
[klevin@cavium-thunderx-login01 stats507f19]$ less hw1.tex  
[less takes up the whole screen]
```

```
This is just a dummy file that I wrote  
as an example.  
An actual tex file wouldn't look like this.  
It would have a bunch of stuff like  
\begin{definition}  
An integer  $p > 1$  is called \emph{prime}  
is its only divisors are  $1$  and  $p$ .  
\end{definition}  
and it would have a preamble  
section declaring its document type  
and a bunch of other stuff.
```

```
hw1.tex (END)
```

**Note:** press “q” to quit `less/more` and return to the command line.

# Basic commands: `mkdir`

`mkdir dirname` : creates a new directory called `dirname`, if it doesn't exist

```
[klevin@cavium-thunderx-login01 stats507f19]$ ls
hw1.tex  hw2.tex  hw3.tex
[klevin@cavium-thunderx-login01 stats507f19]$ mkdir hadoop_stuff
[klevin@cavium-thunderx-login01 stats507f19]$ ls
hadoop_stuff  hw1.tex  hw2.tex  hw3.tex
[klevin@cavium-thunderx-login01 stats507f19]$
```

# Basic commands: mv

`mv file1 file2` : “moves” `file1` to `file2`, overwriting `file2`.

If `file2` is a directory, this places `file1` inside that directory, again replacing any existing file with the same **basename** as `file1`. `/path/to/file/basename.txt`

```
[klevin@cavium-thunderx-login01 stats507f19]$ ls
hadoop_stuff hw1.tex hw2.tex hw3.tex
[klevin@cavium-thunderx-login01 stats507f19]$ mv hw2.tex homework2.tex
[klevin@cavium-thunderx-login01 stats507f19]$ ls
hadoop_stuff homework2.tex hw1.tex hw3.tex
[klevin@cavium-thunderx-login01 stats507f19]$
[klevin@cavium-thunderx-login01 stats507f19]$ mv hw1.tex hadoop_stuff
[klevin@cavium-thunderx-login01 stats507f19]$ ls
hadoop_stuff homework2.tex hw3.tex
[klevin@cavium-thunderx-login01 stats507f19]$ ls hadoop_stuff
hw1.tex
[klevin@cavium-thunderx-login01 stats507f19]$
```

# Basic commands: cp

`cp file1 file2` : similar to `mv`, but creates a copy of `file1` with name `file2`

So `cp` is like `mv` but `file1` is copied instead of being renamed

```
[klevin@cavium-thunderx-login01 stats507f19]$ cat homework2.tex
This is the second homework!
[klevin@cavium-thunderx-login01 stats507f19]$ cp homework2.tex HW2.tex
[klevin@cavium-thunderx-login01 stats507f19]$ cat homework2.tex
This is the second homework!
[klevin@cavium-thunderx-login01 stats507f19]$ cat HW2.tex
This is the second homework!
[klevin@cavium-thunderx-login01 stats507f19]$ ls
hadoop_stuff  homework2.tex  HW2.tex  hw3.tex
```

**Note:** to copy a directory, you must include the `-r` flag to `cp`: `cp -r dirname otherdirname`

## Basic commands: `rm`

`rm filename` : deletes the file `filename`. **Be very very careful with this!**

```
[klevin@cavium-thunderx-login01 stats507f19]$ ls
hadoop_stuff homework2.tex HW2.tex hw3.tex
[klevin@cavium-thunderx-login01 stats507f19]$ rm HW2.tex
[klevin@cavium-thunderx-login01 stats507f19]$ ls
hadoop_stuff homework2.tex hw3.tex
[klevin@cavium-thunderx-login01 stats507f19]$
```

# Basic commands: `logout`

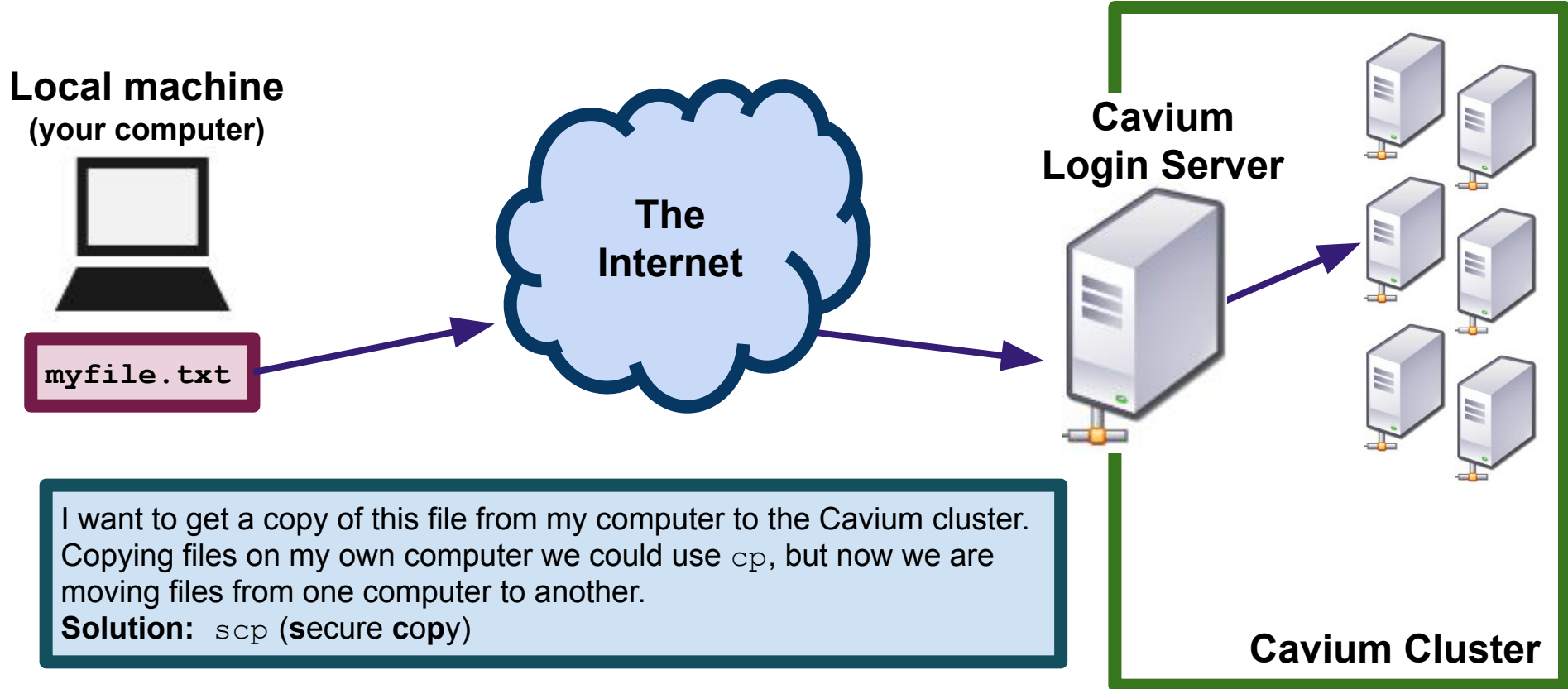
`logout`: close connection to the current machine

```
[klevin@cavium-thunderx-login01 stats507f19]$ logout  
Connection to cavium-thunderx.arc-ts.umich.edu closed.  
keith@Steinhaus:~$
```

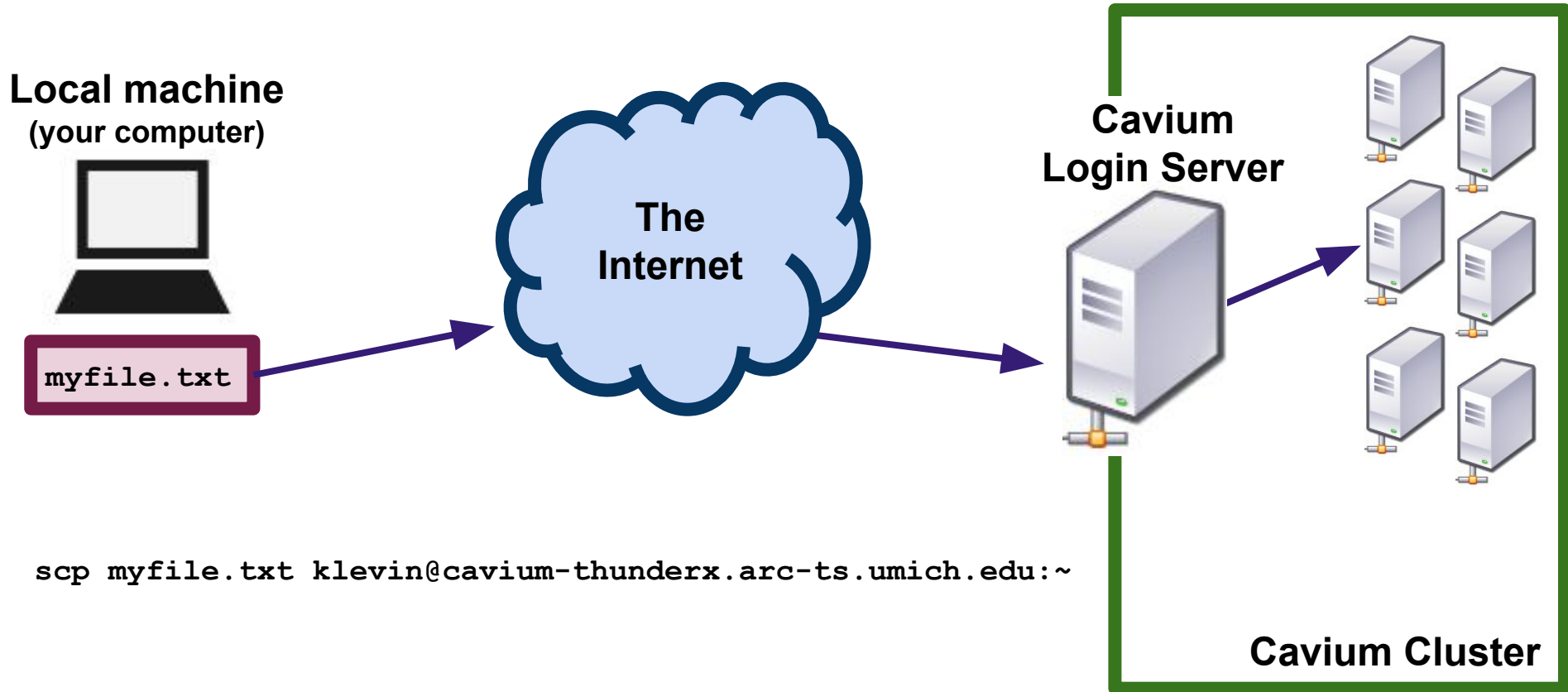
**Note:** depending on the type of shell session in use, you may need to use `exit` or `ctrl-D` to log off.



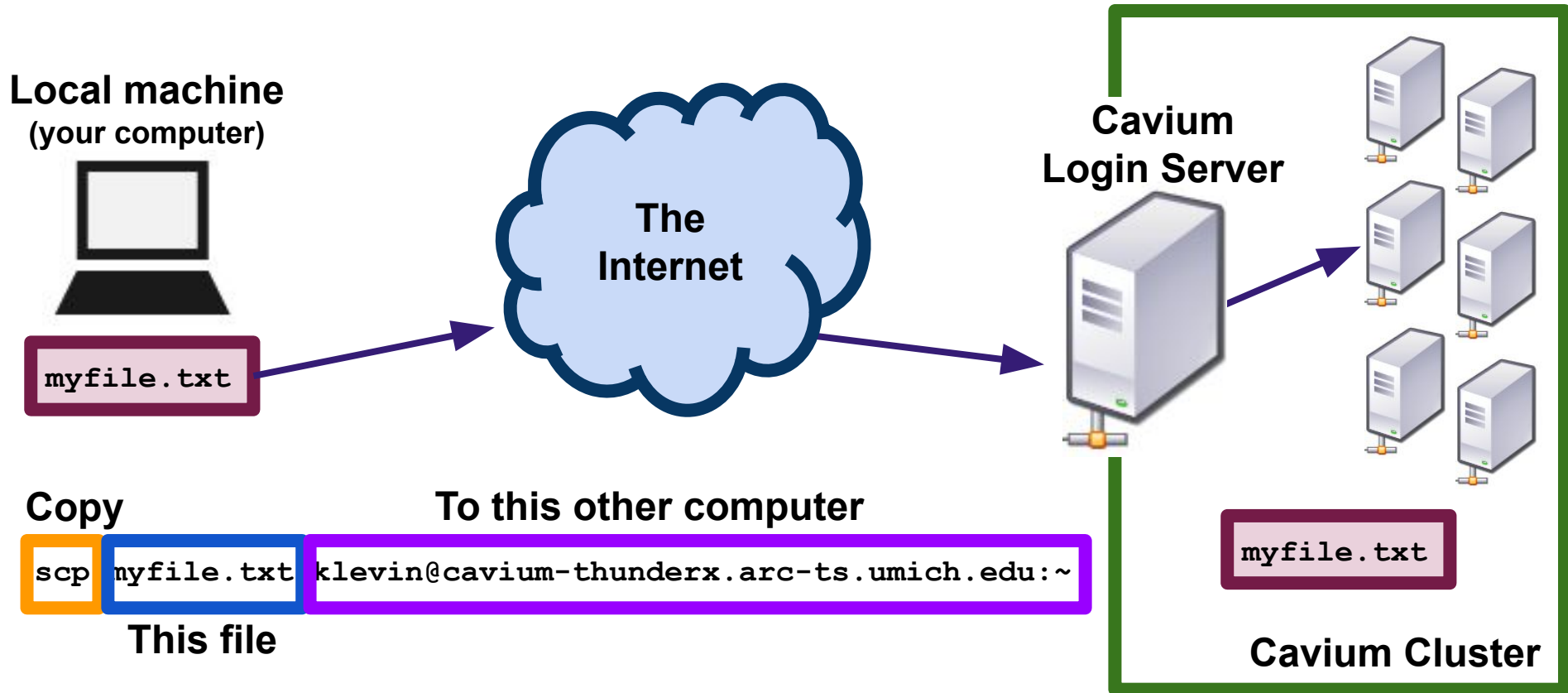
# Moving files between machines: scp (Secure copy)



# Moving files between machines: scp (Secure copy)



# Moving files between machines: scp (Secure copy)

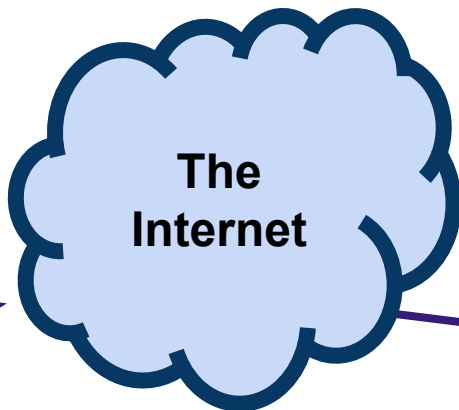


# Moving files between machines: scp (Secure copy)

Local machine  
(your computer)



myfile.txt



The  
Internet

Cavium  
Login Server



myfile.txt

Cavium Cluster

After the colon we specify a path. This is the location where the file will be saved.

```
scp myfile.txt klevin@cavium-thunderx.arc-ts.umich.edu:~
```

# Moving files between machines: scp (Secure copy)

```
scp localfile username@hostname:path/to/file
```

Copy a file from your machine to some other machine via ssh

```
scp username@hostname:path/to/file localfile
```

Copy a file from another machine to your machine via ssh

```
keith@Steinhaus:~$ scp myfile.txt
klevin@cavium-thunderx.arc-ts.umich.edu:~/stats507f19/myfile.txt
Password:
[authentication]
myfile.txt                               100%  14    0.0KB/s   00:00
keith@Steinhaus:~$ ssh -X klevin@cavium-thunderx.arc-ts.umich.edu
Password:
[authentication]
[klevin@cavium-thunderx-login01 ~]$ ls stats507f19/
hadoop_stuff  homework2.tex  hw3.tex  myfile.txt
```

# You will need `scp` for homeworks

If you are on UNIX/Linux/Mac, you can just use `scp` from the command line

If you are on Windows, make sure you have one of:

1. `cygwin` installed and working
2. PuTTY installed with `pscp` working
3. `winscp`

You should try and copy a file to/from Fladoop to make sure everything works

And come talk to me if there are problems!

# We've only scratched the surface!

The UNIX command line is extremely powerful!

Offers numerous tools for working with text and general data wrangling:

`grep, sed, awk, tr, cut, ...`

Ability to use the command line is crucial to being a good “data scientist”

Command line, once you're good at it, makes things very fast

2-3 lines of shell script to do what would take an entire Python program!

# We've only scratched the surface!

The UNIX command line is extremely powerful!

Offers numerous tools for working with text and general data wrangling:

`grep, sed, awk, tr, cut, ...`

Ability to use the command line is crucial to being a good “data scientist”

Command line, once you're good at it, makes things very fast

2-3 lines of shell script to do what would take an entire Python program!

We'll come back to some of these more advanced tools later in the course.