Henzi Kou & Austin Robison
CIS 399: Introduction to System Administration
Steve VanDevender
16 August 2019

Implementing Mail Server in AWS Instances

Summary

The goal of this project is to implement a simple mail transfer agent (MTA) and a mail user agent (MUA). In our case we decided to use Postfix as our MTA to allow users to send and receive mail from our AWS instances via basic terminal SMTP protocol or through an MUA. Any of the created user accounts from the week five assignment are thus able to send emails once they have logged onto our instances. In addition to being able to send messages, our second goal is to allow users to receive email messages with a MUA. We support a couple of Unix based email clients, Mutt and Alpine, that users can choose from to install onto their accounts that will work in unison with Postfix.

Outset Goals

- Implement Postfix using Puppet configurations to monitor and ensure that it is properly installed and running on both of our instances
- Ensure that we are able to send emails in our own personal instances using SMTP protocol on our AWS instances
- Install and implement common MUA clients, Mutt and Alpine, to allow the ability to read and compose emails
- Test both Mutt and Alpine clients are working properly and emails can be composed and read from both clients
- Both instances are able to send receive emails via SMTP or MUA clients

Goals Met

From the outset of the declared goals in our project proposal, it was slightly ambitious of us. Some of the proposed goals were not met, instead we approached the project at the most base level of completion. The final result of the project displays the completion of the most basic email server. We are able to send and receive emails all on our AWS instances and additionally have Steve successfully log onto our instance and compose and send an email to himself, and then ultimately receive it as well. However we were unable to implement an IMAP protocol to allow users remote access to their email accounts.

Effects on User Population & Support Issues

With this project all of the users that have accounts on both of our instances are now able to send and receive emails. Though this service is not as advanced as popular email services like Outlook and Gmail it allows users to manage their emails on their user accounts. Additionally, support issues for users may be somewhat frequent because of the configuration and it difficulty of use compared to the common email services. If issues do arise users can contact us through our emails which are provided at the Resources section.

Security Issues

One security issue that arises is the case of a compromised private key of a user would allow unwanted access on their account. The only way to combat this situation would be the removal of their account and their responsibility to create a new key. Additionally, providing emails for our account users will create a possibility for one individual to send spam emails to the rest of the users. This is because all of our user accounts are publicly posted on GitHub with their account names.

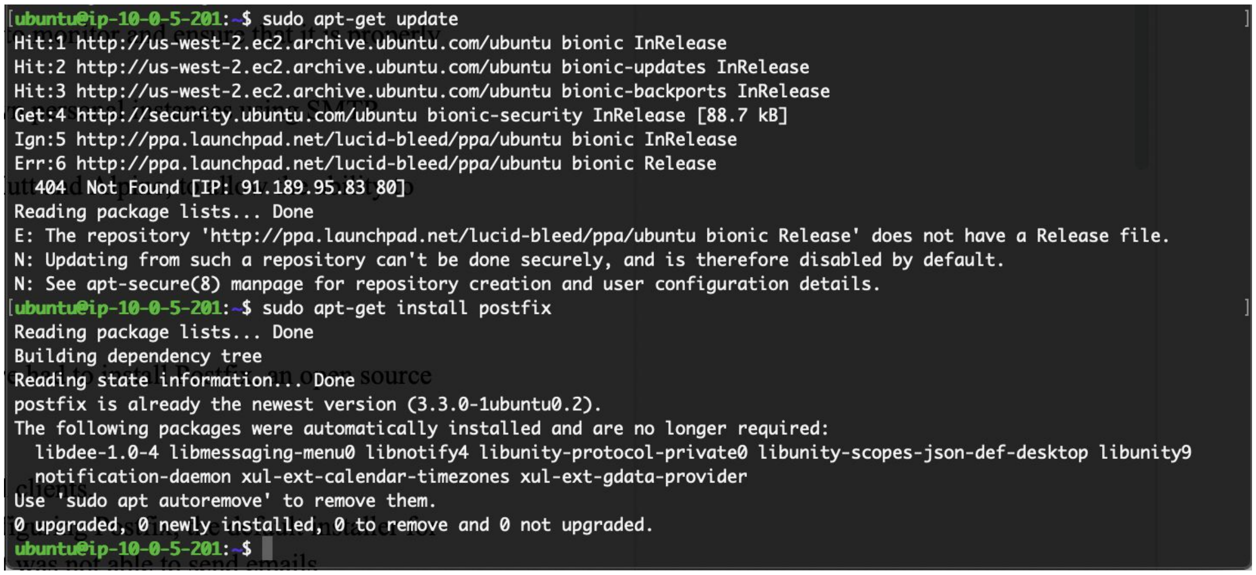
Maintenance & Possible Upgrades

In order to maintain a proper running state of our email service if there are any new users that are added then updates need to be made to our 'User' puppet configuration to create a working account for them. However, to get this project to a more professional state, there are a few things that could be added.

- Configure reverse DNS for instance IP addresses, to keep emails from landing in spam folders of common services like Gmail and the UOregon mail service
- An easier domain name for user input
- Setting up spam filtering with SpamAssassin, to reduce the amount of spam users receive
- Create Puppet configurations to ensure that ~/.muttrc would be pre configured for every user
- Expand our accepted email clients to include various user preferred ones such as Thunderbird

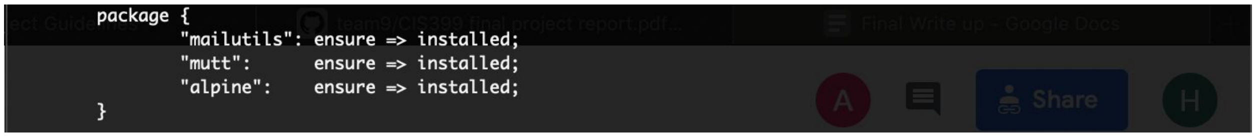
Methodology

To implement email services on our instances we first installed Postfix, an open source mail transfer agent. Before installing we made sure that our instances are up to date, then we performed the installation.



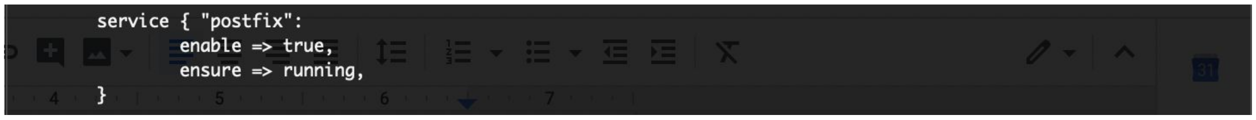
```
ubuntu@ip-10-0-5-201:~$ sudo apt-get update
Hit:1 http://us-west-2.ec2.archive.ubuntu.com/ubuntu bionic InRelease
Hit:2 http://us-west-2.ec2.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:3 http://us-west-2.ec2.archive.ubuntu.com/ubuntu bionic-backports InRelease
Get:4 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Ign:5 http://ppa.launchpad.net/lucid-bleed/ppa/ubuntu bionic InRelease
Err:6 http://ppa.launchpad.net/lucid-bleed/ppa/ubuntu bionic Release
  404  Not Found [IP: 91.189.95.83 80]
Reading package lists... Done
E: The repository 'http://ppa.launchpad.net/lucid-bleed/ppa/ubuntu bionic Release' does not have a Release file.
N: Updating from such a repository can't be done securely, and is therefore disabled by default.
N: See apt-secure(8) manpage for repository creation and user configuration details.
ubuntu@ip-10-0-5-201:~$ sudo apt-get install postfix
Reading package lists... Done
Building dependency tree
Reading state information... Done
postfix is already the newest version (3.3.0-1ubuntu0.2).
The following packages were automatically installed and are no longer required:
  libdee-1.0-4 libmessaging-menu0 libnotify4 libunity-protocol-private0 libunity-scopes-json-def-desktop libunity9
  notification-daemon xul-ext-calendar-timezones xul-ext-gdata-provider
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
ubuntu@ip-10-0-5-201:~$
```

After the installation we composed a Puppet automation job to ensure that everything is properly installed and running on both of our instances. First, we want to ensure that several packages are installed within our instances so that our external functions will be enabled.



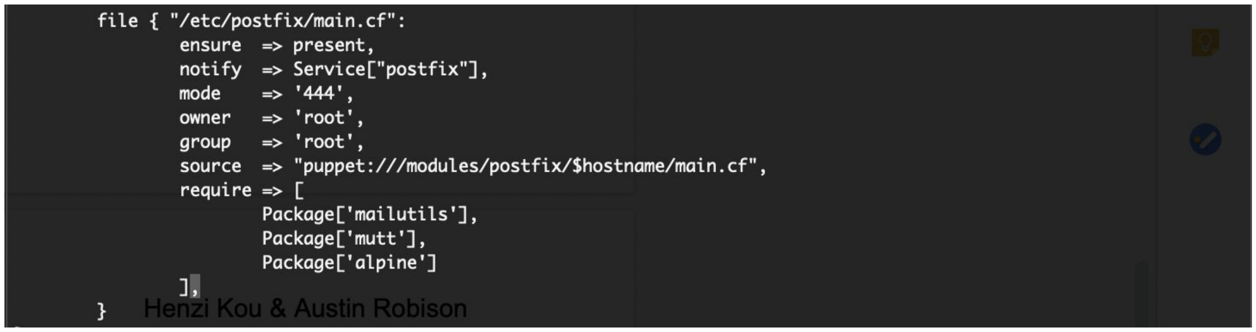
```
        package {
                "mailutils": ensure => installed;
                "mutt":      ensure => installed;
                "alpine":    ensure => installed;
        }
```

Next, we want to always ensure that Postfix is running whenever a user logs into their account on either one of our instances.



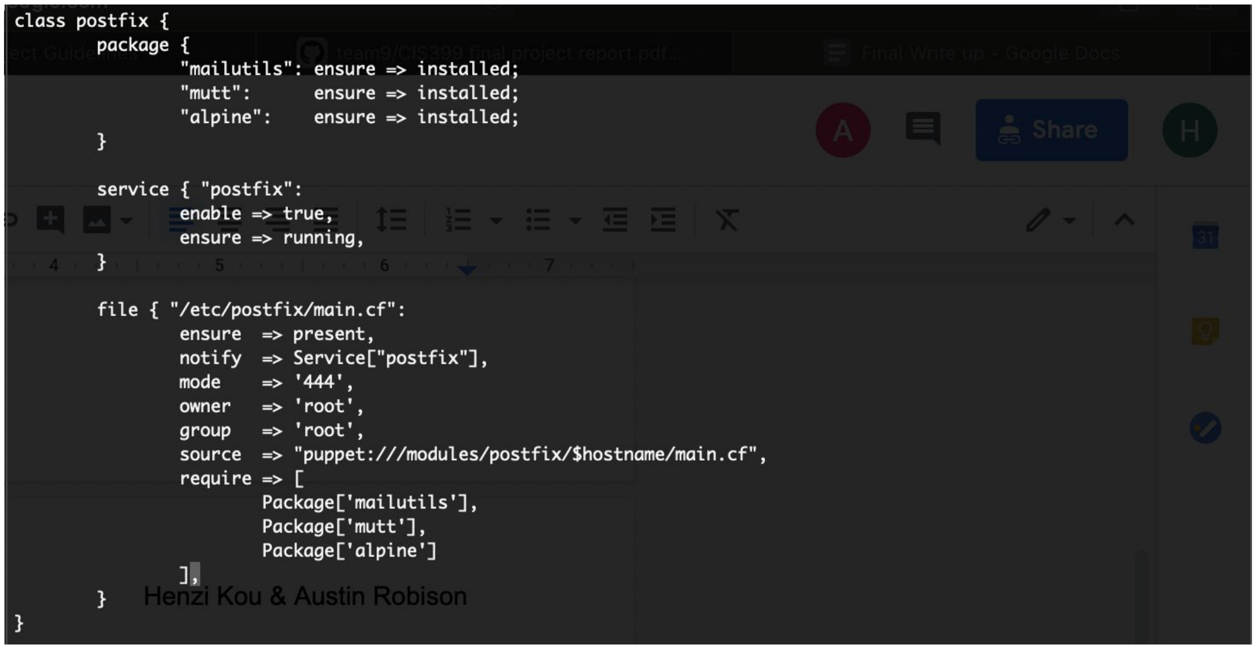
```
        service { "postfix":
                enable => true,
                ensure => running,
        }
```

Finally, we want to ensure that the Postfix config file ‘main.cf’ is located in the correct subdirectory so that Puppet can access its configuration details.



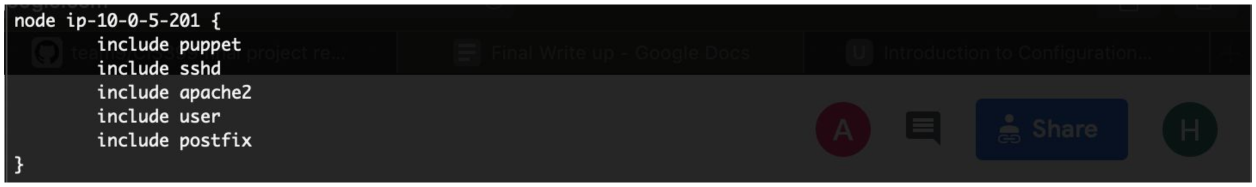
```
        file { "/etc/postfix/main.cf":
                ensure  => present,
                notify  => Service["postfix"],
                mode    => '444',
                owner   => 'root',
                group   => 'root',
                source  => "puppet:///modules/postfix/$hostname/main.cf",
                require => [
                        Package['mailutils'],
                        Package['mutt'],
                        Package['alpine']
                ],
        }
```

This resource ties the previous two resources together by notifying the Postfix service when it is ran and it also ensures that the necessary packages are installed. We additionally create a subdirectory source where Puppet can splice in the local hostname of the user to place the configuration file. The final resulting Puppet configuration file looks like the following,



```
class postfix {
        package {
                "mailutils": ensure => installed;
                "mutt":      ensure => installed;
                "alpine":    ensure => installed;
        }

        service { "postfix":
                enable => true,
                ensure => running,
        }

        file { "/etc/postfix/main.cf":
                ensure  => present,
                notify  => Service["postfix"],
                mode    => '444',
                owner   => 'root',
                group   => 'root',
                source  => "puppet:///modules/postfix/$hostname/main.cf",
                require => [
                        Package['mailutils'],
                        Package['mutt'],
                        Package['alpine']
                ],
        }
}
```

We want these configurations to be present within each instance so we included these into our custom configuration nodes. Below is an example of a node including the Postfix resource.



```
node ip-10-0-5-201 {
        include puppet
        include sshd
        include apache2
        include user
        include postfix
}
```

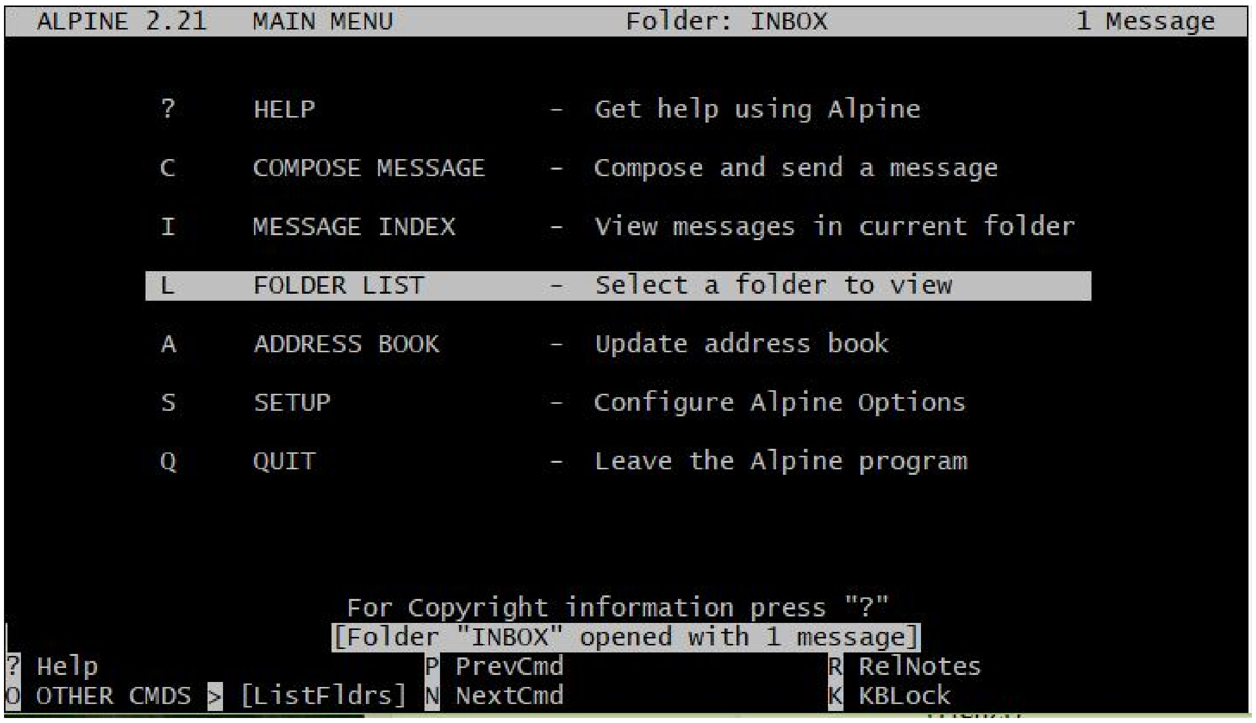
After the resource configurations are created we moved onto the installation of Mutt and Alpine clients onto each of our instances. To install both clients, it follows the same commands as installing Postfix on Ubuntu,

sudo apt-get install mutt
sudo apt-get install alpine

With these two we had to additionally configure Mutt to have the correct send from user address. We will describe later how to properly configure Mutt and Alpine.

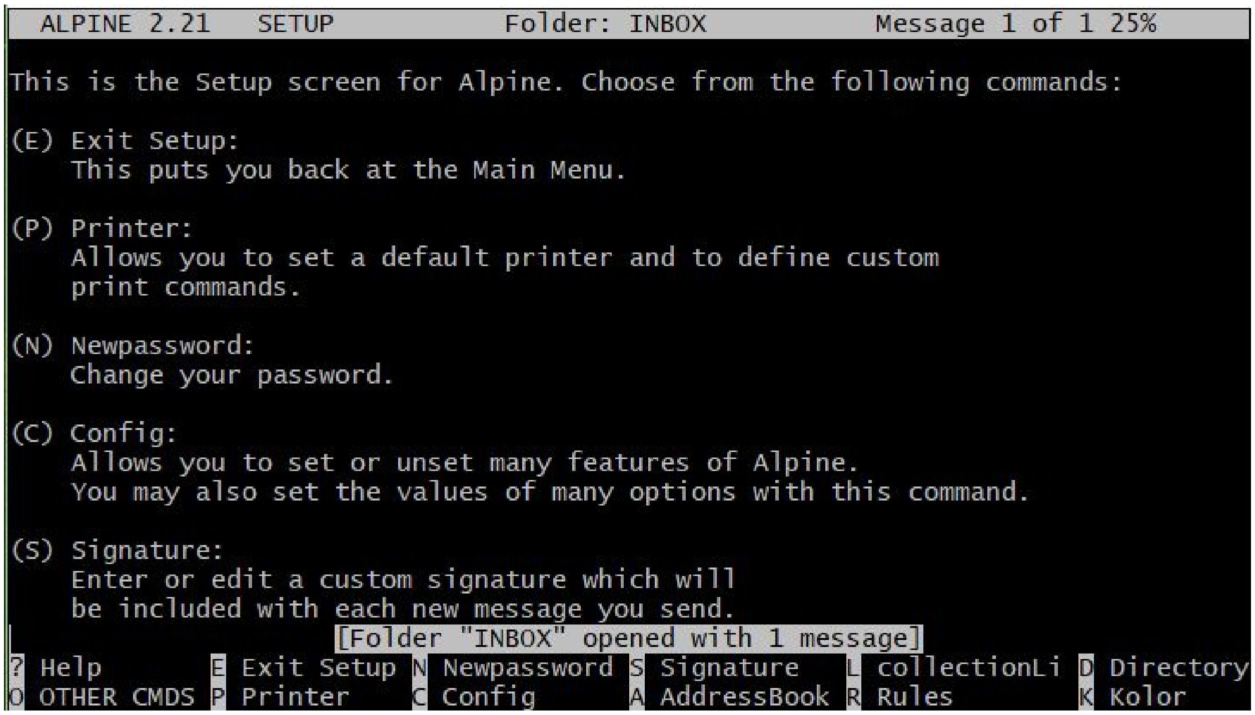
Using Alpine (Easy)

To launch alpine use command: *alpine*. The following should pop up on your screen.



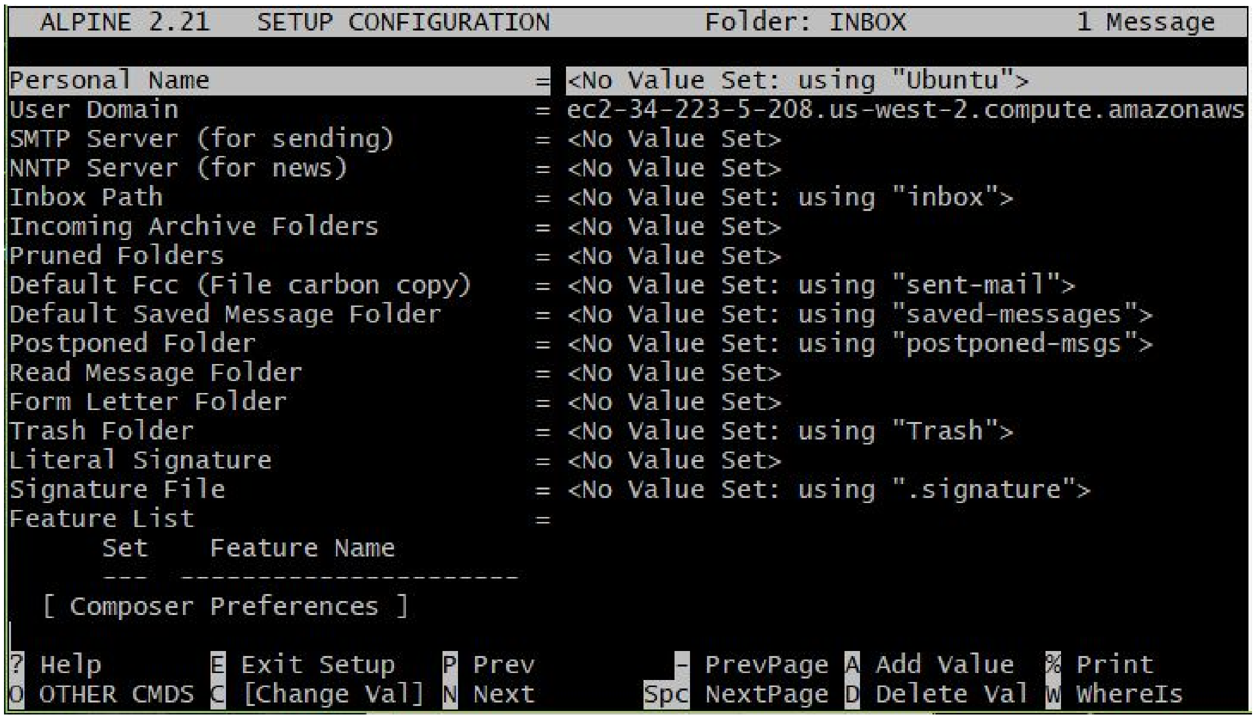
```
  ALPINE 2.21   MAIN MENU              Folder: INBOX              1 Message


          ?     HELP                -  Get help using Alpine

          C     COMPOSE MESSAGE     -  Compose and send a message

          I     MESSAGE INDEX       -  View messages in current folder

          L     FOLDER LIST         -  Select a folder to view

          A     ADDRESS BOOK        -  Update address book

          S     SETUP               -  Configure Alpine Options

          Q     QUIT                -  Leave the Alpine program



                      For Copyright information press "?"
                     [Folder "INBOX" opened with 1 message]
? Help                      P PrevCmd                   R RelNotes
O OTHER CMDS > [ListFldrs]  N NextCmd                   K KBLock
```

Press S to navigate to setup.



```
  ALPINE 2.21   SETUP             Folder: INBOX          Message 1 of 1 25%

This is the Setup screen for Alpine. Choose from the following commands:

(E) Exit Setup:
    This puts you back at the Main Menu.

(P) Printer:
    Allows you to set a default printer and to define custom
    print commands.

(N) Newpassword:
    Change your password.

(C) Config:
    Allows you to set or unset many features of Alpine.
    You may also set the values of many options with this command.

(S) Signature:
    Enter or edit a custom signature which will
    be included with each new message you send.
                     [Folder "INBOX" opened with 1 message]
? Help         E Exit Setup N Newpassword S Signature    L collectionLi D Directory
O OTHER CMDS   P Printer    C Config      A AddressBook  R Rules        K Kolor
```

Then press C to navigate to config.



```
  ALPINE 2.21   SETUP CONFIGURATION          Folder: INBOX          1 Message

Personal Name                       = <No Value Set: using "Ubuntu">
User Domain                         = ec2-34-223-5-208.us-west-2.compute.amazonaws
SMTP Server (for sending)           = <No Value Set>
NNTP Server (for news)              = <No Value Set>
Inbox Path                          = <No Value Set: using "inbox">
Incoming Archive Folders            = <No Value Set>
Pruned Folders                      = <No Value Set>
Default Fcc (File carbon copy)      = <No Value Set: using "sent-mail">
Default Saved Message Folder        = <No Value Set: using "saved-messages">
Postponed Folder                    = <No Value Set: using "postponed-msgs">
Read Message Folder                 = <No Value Set>
Form Letter Folder                  = <No Value Set>
Trash Folder                        = <No Value Set: using "Trash">
Literal Signature                   = <No Value Set>
Signature File                      = <No Value Set: using ".signature">
Feature List                        =
      Set    Feature Name
      ---   ----------------------
  [ Composer Preferences ]

? Help         E Exit Setup    P Prev          - PrevPage A Add Value  % Print
O OTHER CMDS   C [Change Val]  N Next        Spc NextPage D Delete Val W WhereIs
```

Depending on which instance you are logged in to, you will have to set your "user domain" appropriately. (Example for Austin's instance, personal name will be each users home directory by default)

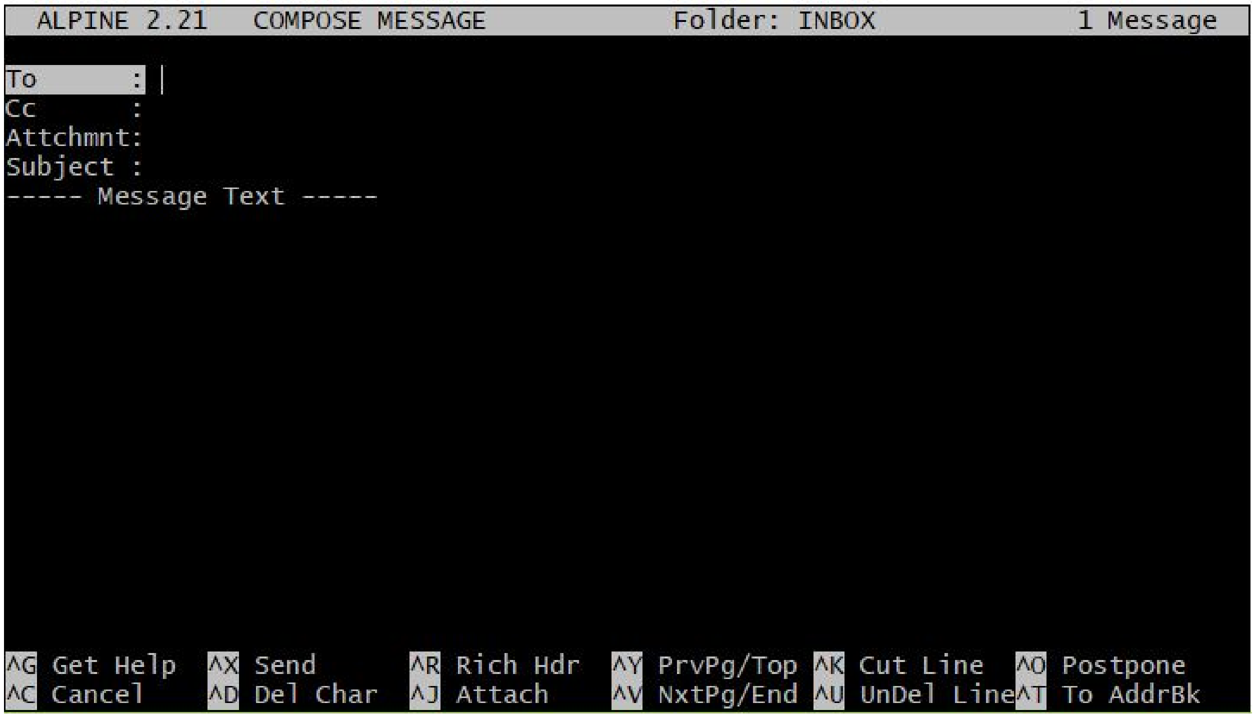
Henzi's Instance:

ec2-54-190-27-30.us-west-2.compute.amazonaws.com

Austin's Instance:

ec2-34-223-5-208.us-west-2.compute.amazonaws.com

Then you can press E to exit setup, and press C to compose a message, the alpine client is rather easy to navigate; after you enter a field, press enter to go to the next field, if a field is empty you can press backspace to navigate up fields, and hit CTRL+X to send (it will ask give you an are you sure message with an y/n answer).



```
  ALPINE 2.21   COMPOSE MESSAGE             Folder: INBOX            1 Message

To      : |
Cc      :
Attchmnt:
Subject :
----- Message Text -----












^G Get Help   ^X Send       ^R Rich Hdr   ^Y PrvPg/Top ^K Cut Line   ^O Postpone
^C Cancel     ^D Del Char   ^J Attach     ^V NxtPg/End ^U UnDel Line^T To AddrBk
```

Reading Emails Using Alpine

In the main menu, you can press I to view a list of your emails, pressing enter to open an email and < to navigate back to your list of emails and to go back to the main menu

Using Mutt

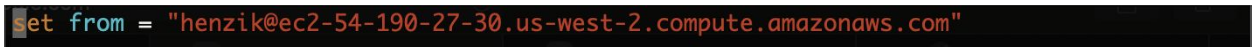
Mutt Configuration:

In your user directory, type the following command: *vi ~/.muttrc* (note that you can use any text editor of your choice, we decided to use vi/vim for simplicity). This file will be empty at first, you only need need to enter one line depending on what instance you are using:

Henzi's Instance
set from = "<YOUR_USERNAME>@ec2-54-190-27-30.us-west-2.compute.amazonaws.com"

Austin's Instance
set from = "<YOUR_USERNAME>@ec2-34-223-5-208.us-west-2.compute.amazonaws.com"

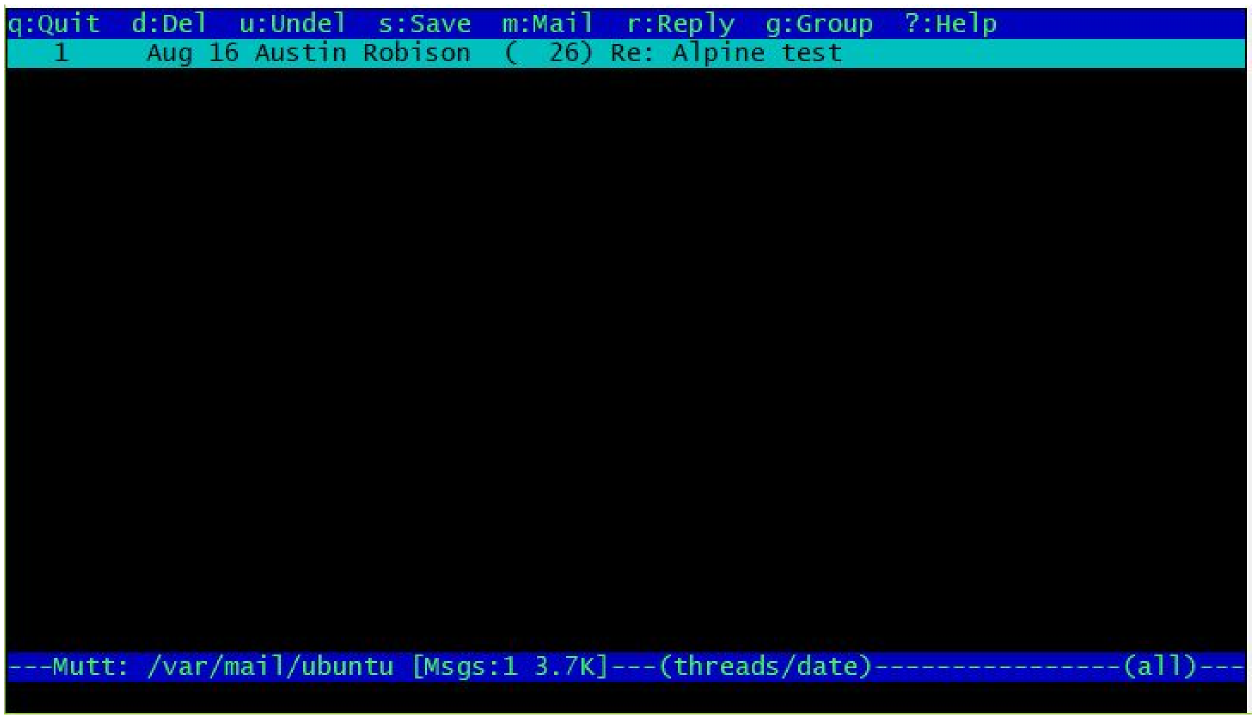
And ":wq" to save your config file if you are using vi/vim. An example *~/.muttrc* configuration is shown below.



```
set from = "henzik@ec2-54-190-27-30.us-west-2.compute.amazonaws.com"
```

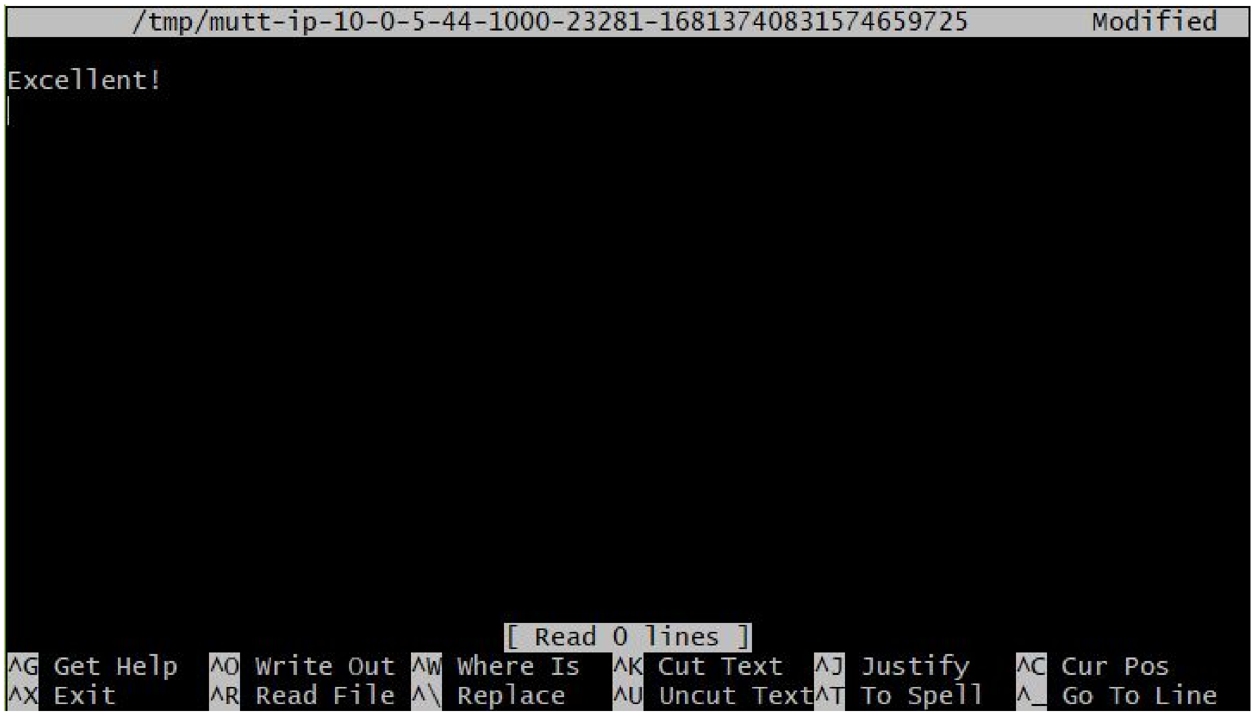
Sending Emails Using Mutt:

Use command *mutt*. Then the following should pop up on your screen.



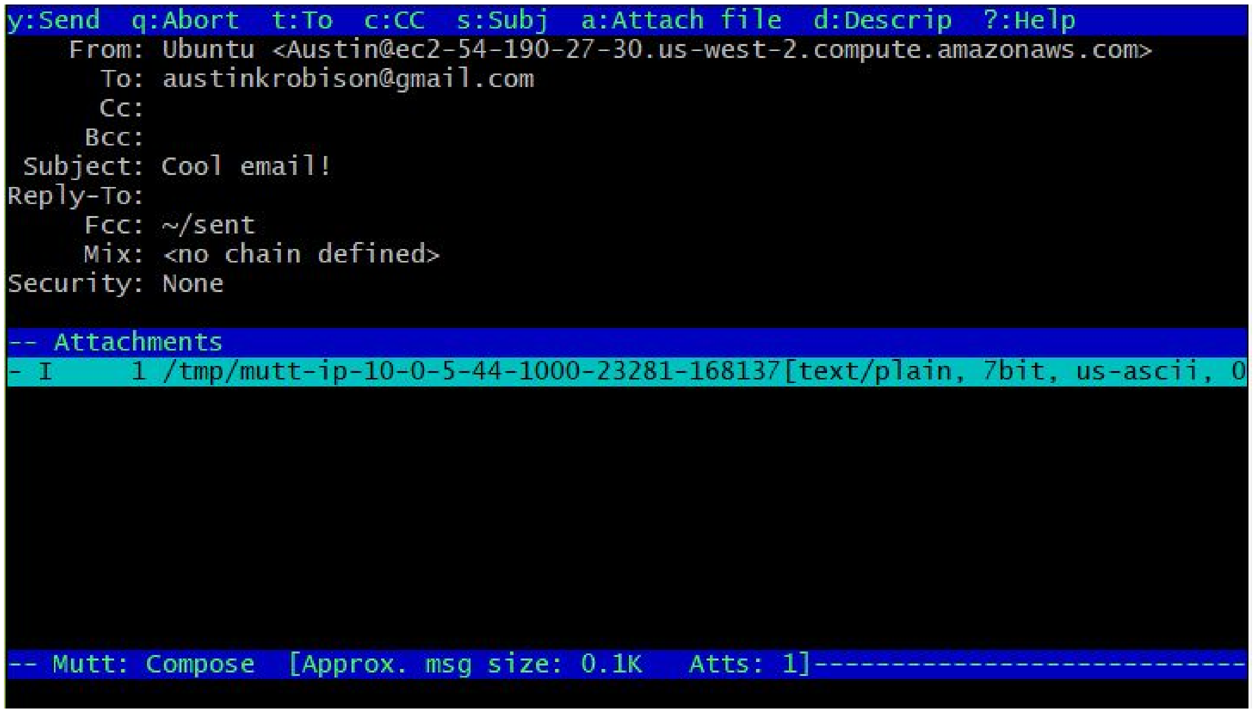
```
q:Quit  d:Del  u:Undel  s:Save  m:Mail  r:Reply  g:Group  ?:Help
   1      Aug 16 Austin Robison  (  26) Re: Alpine test















---Mutt: /var/mail/ubuntu [Msgs:1 3.7K]---(threads/date)----------------(all)---
```

Press M to start composing an email, at the very bottom of the terminal you will see *to:* and *subject:* and enter their fields accordingly. Next you enter the contents of the email, CTRL+X to write it, CTRL+X to exit the screen,



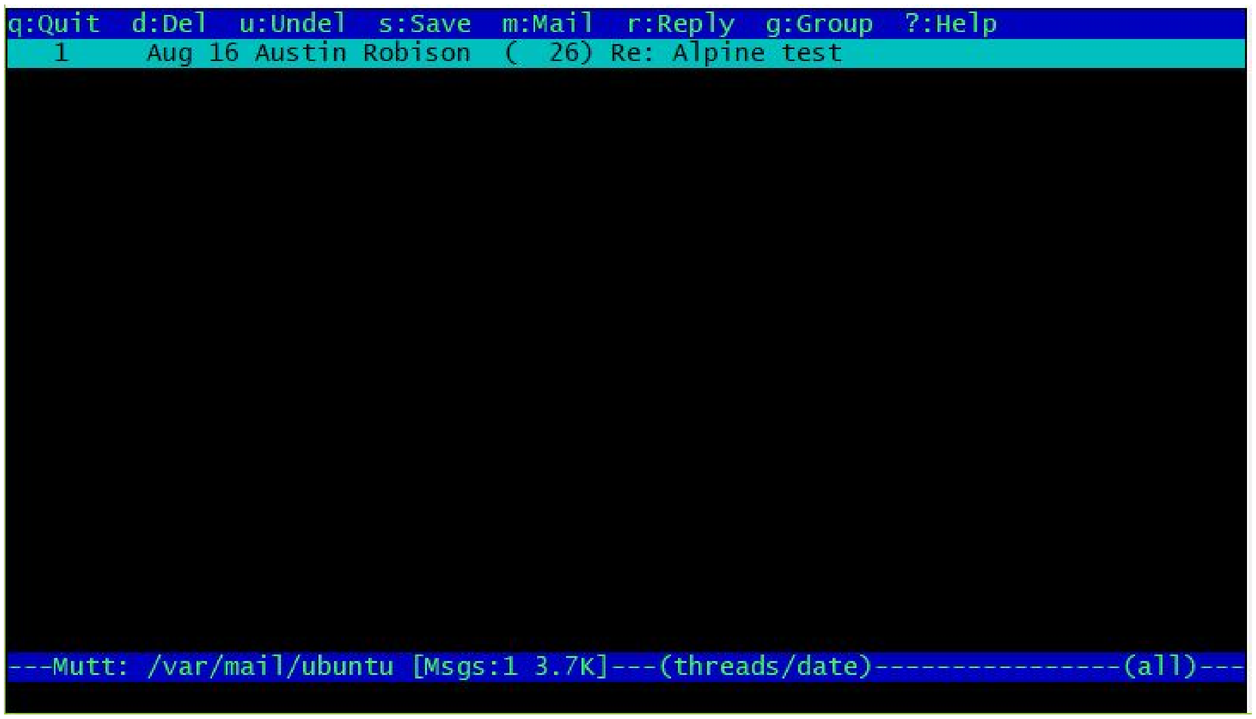
```
         /tmp/mutt-ip-10-0-5-44-1000-23281-16813740831574659725       Modified

Excellent!




                                [ Read 0 lines ]
^G Get Help   ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit       ^R Read File ^\ Replace   ^U Uncut Text^T To Spell  ^_ Go To Line
```

Press enter to see a summary or make last minute changes, then Y to send.



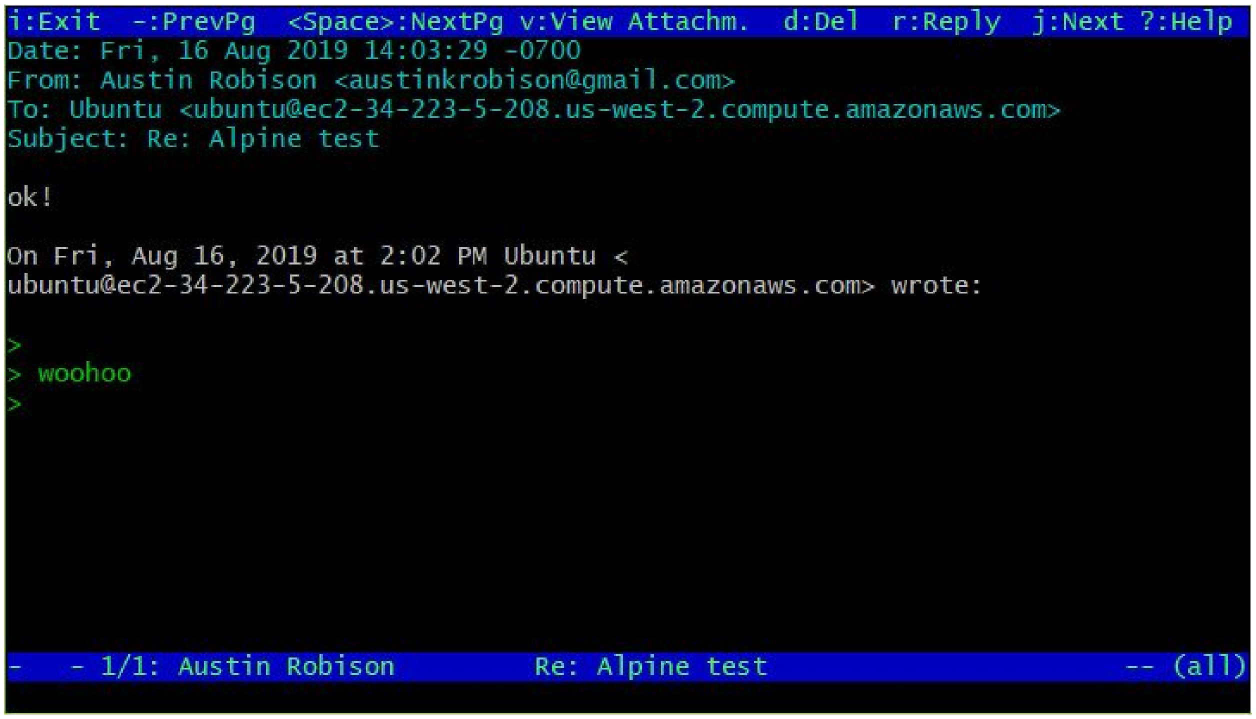
```
y:Send  q:Abort  t:To  c:CC  s:Subj  a:Attach file  d:Descrip  ?:Help
    From: Ubuntu <Austin@ec2-54-190-27-30.us-west-2.compute.amazonaws.com>
      To: austinkrobison@gmail.com
      Cc:
     Bcc:
 Subject: Cool email!
Reply-To:
     Fcc: ~/sent
     Mix: <no chain defined>
Security: None

-- Attachments
- I     1 /tmp/mutt-ip-10-0-5-44-1000-23281-168137[text/plain, 7bit, us-ascii, 0



-- Mutt: Compose  [Approx. msg size: 0.1K   Atts: 1]---------------------------
```

Reading Emails Using Mutt;
Use command: *mutt*. This displays your inbox



```
q:Quit  d:Del  u:Undel  s:Save  m:Mail  r:Reply  g:Group  ?:Help
   1      Aug 16 Austin Robison  (  26) Re: Alpine test



---Mutt: /var/mail/ubuntu [Msgs:1 3.7K]---(threads/date)----------------(all)---
```

You can view an email by highlighting it and pressing ENTER.



```
i:Exit  -:PrevPg  <Space>:NextPg v:View Attachm.  d:Del  r:Reply  j:Next ?:Help
Date: Fri, 16 Aug 2019 14:03:29 -0700
From: Austin Robison <austinkrobison@gmail.com>
To: Ubuntu <ubuntu@ec2-34-223-5-208.us-west-2.compute.amazonaws.com>
Subject: Re: Alpine test

ok!

On Fri, Aug 16, 2019 at 2:02 PM Ubuntu <
ubuntu@ec2-34-223-5-208.us-west-2.compute.amazonaws.com> wrote:

>
> woohoo
>

-   - 1/1: Austin Robison         Re: Alpine test                        -- (all)
```

Documentation

The following links provide in-depth guides and the documentation for each of the resources that were utilized within our project.

- https://wiki.ubuntu.com/Mutt
- http://www.postfix.org
- http://manpages.ubuntu.com/manpages/trusty/man1/alpine.1.html

All of our Puppet modules and resource configurations can be found on our GitHub repository, defaultName-puppet for the user community for reference and our personal administrative reference.

Resources:

Contacts:

- Austin: austinkrobison@gmail.com
- Henzi: henzik@cs.uoregon.edu

GitHub Repositories:

- https://github.com/cis399-2019-team/defaultName-puppet
- https://github.com/cis399-2019-team/defaultName