"Dainty Ditty's for Astro-Informatic Dockers"

Docker Containers

Astrometry.net

Sextractor / PSFEx / SCAMP / SWARP

Astropy and Custom Python

PostgreSQL and Q3C

Texlive

Sphinx

Local repositories / conda channels / and other containerized tools

Wayne Green

Saturday 1st January, 2022

Copyrights and Trademarks:

Arduino 2022, under Creative Commons licensing; Anaconda is a collection of Python and analytic tools by Anaconda® Inc.; Apple® , iOS® , OS/X® and Macbook® are a trademarks of Apple Inc., registered in the U.S. and other countries; Astrometry.net Copyright 2006-2010 Michael Blanton, David W. Hogg, Dustin Lang, Keir Mierle and Sam Roweis, Copyright 2011-2013 Dustin Lang and David W. Hogg. Copyright 2006-2022 Michael Blanton, David W. Hogg, Dustin Lang, Keir Mierle and Sam Roweis [11] (the Astrometry.net Team); The Astropy Project [13] is a loose-confederation funded by NumFOCUS, Inc.; Docker, Inc. provides the technology for Docker Environments, Containers etc, ©2013-2015 Docker, Inc. All rights reserved; IRAF, PyRAF and Ureka Copyright ©2018 Association of Universities for Research in Astronomy (AURA); Bokeh,2021 Bokeh Contributors, is a fiscally sponsored project of NumFOCUS; Django The Django Software Foundation; NumPy ©2005-2018, NumPy Developers ; The "Math Kernel Library" (MKL®) and Intel® are registered by Intel Corporation – Santa Clara, CA, USA; Microsoft, "Microsoft Windows", WSL are registered trademark of Microsoft Corporation; PostgreSQL® is a registered trademark of PostgreSQL Global Development Group; Python is Copyright ©2001-2018 Python Software Foundation. All rights reserved ; Raspberry Pi, governed by the Raspberry Pi Foundation with offerings covered by the Creative Commons CC BY-SA licensing; SAOImage/ds9 and code therein is licensed through the Smithsonian Astrophysical Observatory; SExtractor [1] (Emmanuel Bertin, author) copyright ©2010 IAP - CNRS / Universite P.&M.Curie ; Ubuntu ©2015 Canonical Ltd. Ubuntu and Canonical are registered trademarks of Canonical Ltd; UNIX® is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Ltd; Linux® is the registered trademark of Linux Mark Institute, on behalf of Linus Torvalds, in the U.S. and other countries.

Contents

List of Figures

Preface

This document contains notes about building a photometry pipeline in a docker environment for Linux and Win10.

First and foremost “Docker Containers” (containers) are “designed” for developers, used for “service deployment” and (while desirable), not for “application deployment”.

In the context of this article, a certain vocabulary is created to help avoid confusion – especially with overloaded terms:

PMachine - the Physical machine you can touch, that holds logical containers. The OS is usually referred to as the host os.

CMachine - a ’Container’ with its view of a ’OS’. In short most of the time CMachines think they are PMachines – same commands, same network issues, etc.

X/11 - The mainstay of the Unix windowing system that consists of 1) clients – programs creating output for display, buttons to push, image frames to show graphs etc and 2) servers – the logic that accepts those commands and actually “renders the content” to a ’screen’. The sense of CLIENT and SERVER is backwards to that of a database. X11 is old, robust, made to support graphics in the times of dial-up modems and is very light-weight in its bandwidth overhead. It is NOT your great-grandson’s Wayland – while still pending for Ubuntu 20.04, support is still weak.

Docker extends the host’s hardware which is simply a “Physical Machine” (PMachine) with modern hyper-visor support.

A PMachine runs a container that “believes” it is a physical machine. A container, herein, is refereed to as a “Container Machine” (CMachine). Remember, PMachines have atoms and CMachines have electrons. A container is managed by a daemon that uses hypervisor functionality to ’glue’ the OS within the container (CMachine) to its ’host’ (PMachine).

A container is an empty light-weight hypervisor interface that is filled first with a Linux based OS then filled with any user code.

```
here:me> docker export <CONTAINER ID> > /home/export.tar
here:me> scp /home/export.tar them@there.net:/home/them
there:them> cat /home/export.tar | docker import - some-name:latest
```

The argument over who owns the PMachine’s GUI user experience becomes, well, quite territorial when it comes to hosting mainstay

X-Windows environment in a Microsoft OS or in Apple iOS's diverging OS. You may be better served with a VBox[1]. Each camp can make a compelling argument to the user while avoiding the user's real need – to express their problem to software while the 'interface' is invisible. To take the user's focus away from their problem is, well not a Good Thing(™).

The learning curve for this task is as straight-forward as stepping of a high cliff. Today, Saturday 1st January, 2022, a decent task it to encapsulate a body of 'codes'[2] that are critical to science and somewhat lost to history.

ACTION:
Flesh
hypevisors

[1] Oracle Corp. VirtualBox or other vendors equivalents.

[2] I'm an astronomer with a storied career in computing.

1 Important Details

These notes are from a Linux perspective. There are two openings

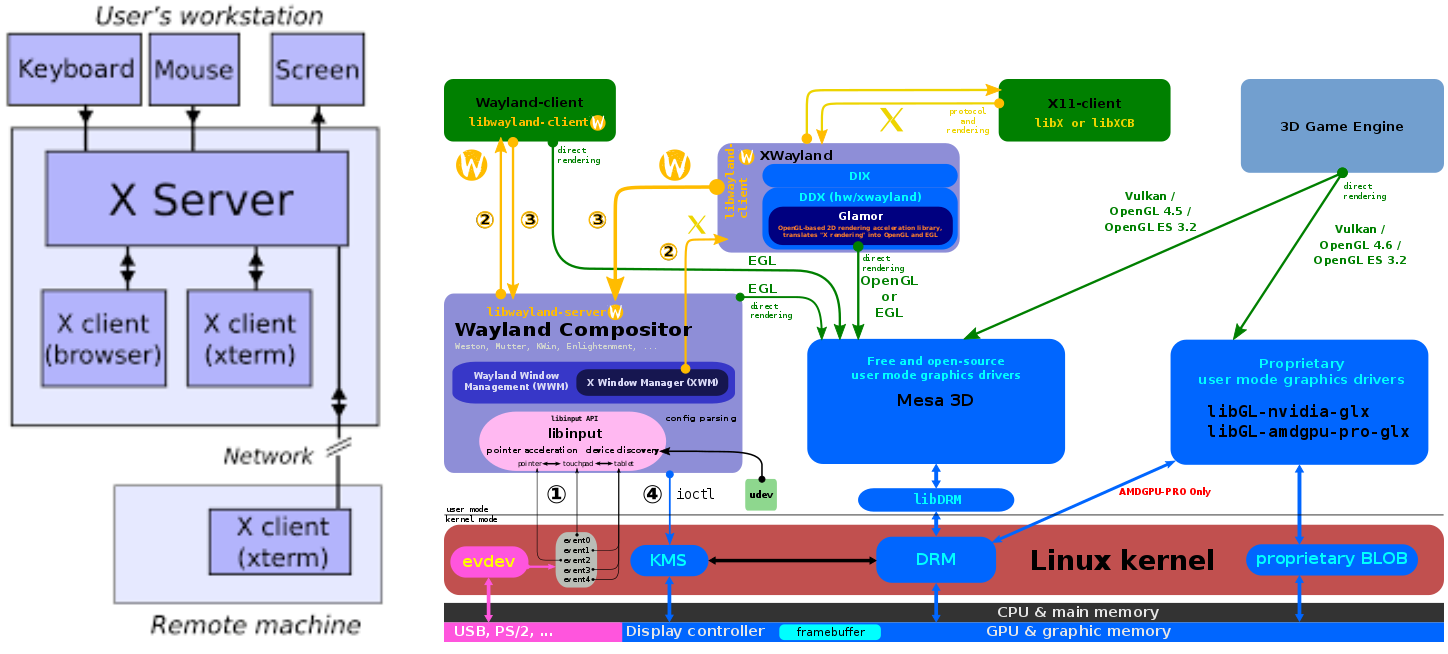
When you use the container, it's transient (current) state will not "persist"; therefore any newly created files inside its file system are lost. THE STATE IS LOST WHEN THE CONTAINER STOPS! That is a Good Thing(™)... you always start fresh.

You can "pause" and "commit" the running container with a new tag/version. Pick up where you leave off. This is handy as checkpoints for containers running in production – where one may restore to a recent/earlier state.

Cast of characters: the $USER is Bob Brown (herein bob), with /home/bob as the $HOME directory.

ACTION:
perspectiv

There are massive confusions surrounding the terms “host”, “client” and “server”. These are overloaded terms. Again, for clarity, the term PMachine will refer to the physical machine controlling your Desktop, and the term CMachine represents the<> “containerized” machine.



(a) X11 (b) Wayland

Figure 1-1: The reversed sense of ’server’ and client’ in X-Windows.

(a) David Gerard (b) https://commons.wikimedia.org/wiki/User:ScotXW

W.r.t. X11, the term “server” is the OS’s GIO window that “serves” the information from a program (X “client”) to your eyeballs. The “client” does the heavy-lifting/computations to producing the text/images. In normal computing the term “Server” refers to the program/hardware doing the computations and producing the text/images and the “Client” some task or person utilizing the service.

Unix OSs perceive themselves to be “peers”, so the whole Host/Client thing is relative.

The user-level “Docker Desktop” has to be installed on your PMachine:

https://www.docker.com/get-started

On your PMachine: Create a directory called $HOME/Astro.

Under this directory you should put everything containers need access. This is all the fits files, etc. It is the “link” between the CMachine and that part of your PMachine.

A few additional key files are supplied:

A few critical aliases:

ACTION:
Additiona
Software[3]

2 Astrometry.net

Github has a working image of the file.

3 Networking

Astronomy applications use SAMP and XPA. Programs like the Virtual Observatory (VO) [5], Aladin [3], SAOImage/ds9 [10], TOPCAT/ADQL [16, 17], SPLAT-VO [19] readily inter-operate with PMachine/CMachine databases like PostgreSQL. This requires careful management of TCP/IP Port access, the docker bridge and the local PMachine IPTables, etc.

Docker runs containers within a docker bridge – this requires some fancy dancing to get a connection between the PMachine and a CMachine.

```
https://docs.docker.com/engine/reference/commandline/attach/
https://runnable.com/docker/basic-docker-networking

# good youtubes on networking
https://www.youtube.com/watch?v=Yr6-2ddhLVo
https://cntnr.io/running-guis-with-docker-on-mac-os-x-a14df6a76efc  ***
```

A Overview

Docker “Containers” contain, initialize and run a Docker “Image”.

A Docker images is built from a “Dockerfile” using docker build ....

A “Container” can be created from a “Image” using docker create ....

```
Observations
├── /ddMMMyyyy/ # local dates data
│   ├── usw # Nights collection
│   │   ├── reduce.cl # template
│   │   └── reduce.pyraf # template
│   ├── RawData # Nights Raw Data
│   ├── PreAnalysis # The basic work
│   └── Analysis # Non-pipeline work for pre-publication
└── usw # Global templates etc
    ├── Targets # Global target collection usws
    │   └── <target>
    │       ├── hints.ans
    │       ├── cfg.ans
    │       ├── target.tsv
    │       ├── finder.png
    │       ├── Catalog_AAVSO.tsv
    │       ├── Catalog_UCAC4.tsv
    │       └── Catalog_REFCAT2.tsv
```

Layout of the the Observations Environment. The collection of all the observations, general aspects of a user's environment like ds9 fiture templates etc; each target (defining the location and header details), each night observations – their rawdata, preanalysis (augmenting headers, bias/dark/flats, specific reduce.cl; etc), and any analysis prior to publication. Should be under revision management.

B Deployment

The first container to make is the Dockerfile.xeyes container test on the local PMachine that supports X11, then

```
docker run -v /tmp/.X11-unix:/tmp/.X11-unix  \
  -e DISPLAY=$DISPLAY  \
  -h $HOSTNAME  \
  -v $HOME/.Xauthority:/home/lyonn/.Xauthority  \
  --name xeyestest \
  ubuntu_xeyes:1
```

while it is running:

```
docker export xeyestest | gzip > ubuntu_xeyes_1.tar.gz
```

The ubuntu_xeyes_1.tar.gz file is then copied to the 'other' PMachine. For Win10

C Docker Security

Docker interactions are centered around a daemon dockerd [3].

Docker relies heavily on Unix Namespaces, its own TCP/IP stack and bridge arrangements, Unix sockets, and the TCP/IP stack of the machine to facilitate communications between the PMachine and the CMachine via the docker daemon. This should be restricted to local host 127.0.0.1. By default it uses port 2345 for insecure transactions and port 2346 for secure transactions. Other tricks using Certificates of Authority (CA) etc help to further isolate PMachine/CMachine interface and CMachine/CMachine interoperability.

Starting services: systemd vs systemV – Ubuntu uses systemd, alpine uses systemV.

ACTION: flesh systemd[4]

https://towardsdatascience.com/analyzing-docker-image-security-ed5cf7e93751

[3] https://docs.docker.com/engine/reference/commandline/dockerd/

D X11

```
PMachine needs ‘‘X11FORWARDING yes’’ is \dhl{/etc/ssh\_config}
Assure \dhl{/etc/ssh_config}:
X11Forwarding yes
X11DisplayOffset 10
X11UseLocalhost no
sudo service ssh restart
```

https://askubuntu.com/questions/61690/ssh-x-xt-error-cant-open-display-0-0 supplies obscure background config sources.

Where DISPLAY=0 or DISPLAY=0.0 means the 'current' window of a set of virtual/physical displays is in use. Make sure DISPLAY is not being set in /etc/environment or /etc/bash.bashrc, or /.bash.rc or any alias files it may ingest. Run env | grep DISPLAY

The command ssh -X user@machine env will show "environment" things there.

E Local Docker Development Machine

If on a slow network, then create local repo.

```
apt-get install dpkg-dev
cd /opt
mdkir -p /opt/debrepo  # our local depo
cd /opt/debrepo
cp -p /var/cache/apt/archives/*deb .  # get the packages on HOST here.
dpkg-scanpackages . /dev/null | gzip -9c > Packages.gz
```

F PyRAF

PyRAF within a docker container wants to interact with many support programs via XPA and SAMP.

Currently the SAMP and 'message of the day' parts of login.cl don't allow PyRAF to run. Comment them out.

G SAOImage/ds9

SAOImage/ds9 is an enhanced replacement for the old IRAF ximtool – itself a replacement for IRAF' tv/display. It implements XPA (Section **??**). Tcl, and toolkit Tk is a dynamic, late-binding (ie self-modifying) language that, while 'born of frustration' is frustrating to use owing to a few missed tricks, complicated syntax and complicated approach to introspection. That said, users' may extend ds9 in wondrous ways. .

H SAMP

SAMP runs on port 21012. https://localhost:21012 is there! [15, 14]. The snippet in Figure H-2 outlines a quick use of Linux utilities to find the host for the hub. It implements a Unix socket connection using traditional XML-RPC tricks.

```
sudo netstat -lp | grep 21012
tcp6       0      0 [::]:21012    [::]:*   LISTEN      15583/java

ps aux | grep 15583
wayne    15583  0.7  2.0 13649168 686824 pts/1 Sl...
... Apr02 155:01 java -Duk.ac.starlink.topcat.cmdname=topcat...
... -classpath ./../lib/topcat/topcat.jar:/home/wayne/Configuration//postgresql- ...
... 42.2.8.jar uk.ac.starlink.topcat.Driver
```

Figure H-2: SAMP - Example to find the app listening on port 15583.

ACTION: Box fig text[5]

I XPA

XPA like SAMP is a suite of interprocess communications tools. You can obtain the source code from https://github.com/ericmandel/xpa.git, or install with the apt-get xpa-tools (and other language dependent libraries). The GitHub repository has

some interesting python tricks. XPA has ties to tcl making and is the underpinning of the communications between IRAF/PyRAF and ds9.

ACTION: Make X docker exercise[6]

```
me> xpaget xpans
XPA$ERROR no 'xpaget' access points match template: xpans
me >ds9 &
xpaget xpans
DS9 ds9 gs 7f000001:42049 me
```

The port/server/connections were established when ds9 started: 7f000001 string is 127.0.0.1 or localhost is 42049 and the 'user' is me is the port that was established when ds9 started.

XPA has a simple command-line vocabulary that enables a script (bash/python) to make simple queries in both directions. Examples for ds9, while in tcl can be easily transliterated to bash, are at: http://ds9.si.edu/doc/ref/xpa.html .

DS9 ds9 gs 7f000001:42049 me

J bash

Debian uses a nasty /bin/dash in lieu of bash. So,

```
mv /bin/sh /bin/dashsh
ln -s /bin/bash /bin/sh
```

A clean continuumio/anaconda3 was downloaded, and the packages copied from the container. This is a rather large directory.

The recipe for IRAF was run:

```
conda create -n iraf27 python=2.7 iraf-all pyraf-all stsci
conda install -c conda-forge jupyter_contrib_nbextensions
```

... and the resulting packages copied.

A small python script:

```
from collections import OrderedDict()
d = OrderedDict()
f = open('pre','r')
x = open('post','r')
for c in f:
   if(c not in d):
      d[c] = []
   d[c].append('pre  '+c.strip())

for c in x:
   if(c not in d):
      d[c] = []
   d[c].append('post '+c.strip())

with open("postpackges.txt",'w') as o:
   for k,v in d.items():
      if(len(v) == 1):
         print("{}".format(v[0]),file=o)
```

A list of places where there was only 1 entry was created. These were all 'post' by the logic above.

```
cd post
mkdir local-channel
cd local-channel
mkdir linux-64 osx-64
cd linux
cp ../../*.bz2 .
conda index .
#check Note: the two slashes with the file, and --offiline
conda search  --offline -c file://home/wayne/play/bob/oc/gaoiraf iraf-x11 | grep gaoiraf

docker run -it -v /home/wayne/play/bob/oc/gaoiraf:/opt/gaoiraf continuumio/cuupdated
conda config --add channels file://opt/gaoiraf
conda create -c gaoiraf -n iraf27 python=2.7 iraf-all pyraf-all stsci
```

K Ubuntu Packages

Using docker pull ubuntu18.04:latest, build a basic container, adding a few packages. The file /etc/apt/apt.conf.d contains several 'rules' for apt to follow.

For the basic container:

```
### 01-vendor-ubuntu
Acquire::Changelogs::AlwaysOnline "true";
### 01autoremove
APT
{
  NeverAutoRemove
  {
"^firmware-linux.*";
"^linux-firmware$";
"^linux-image-[a-z0-9]*$";
"^linux-image-[a-z0-9]*-[a-z0-9]*$";
  };

  VersionedKernelPackages
  {
# linux kernels
"linux-image";
"linux-headers";
"linux-image-extra";
"linux-modules";
"linux-modules-extra";
"linux-signed-image";
"linux-image-unsigned";
# kfreebsd kernels
"kfreebsd-image";
"kfreebsd-headers";
# hurd kernels
"gnumach-image";
# (out-of-tree) modules
".*-modules";
".*-kernel";
"linux-backports-modules-.*";
"linux-modules-.*";
        # tools
        "linux-tools";
        "linux-cloud-tools";
# build info
"linux-buildinfo";
```

```
# source code
"linux-source";
  };

  Never-MarkAuto-Sections
  {
"metapackages";
"contrib/metapackages";
"non-free/metapackages";
"restricted/metapackages";
"universe/metapackages";
"multiverse/metapackages";
  };

  Move-Autobit-Sections
  {
"oldlibs";
"contrib/oldlibs";
"non-free/oldlibs";
"restricted/oldlibs";
"universe/oldlibs";
"multiverse/oldlibs";
  };
};
### 01autoremove-kernels
// DO NOT EDIT! File autogenerated by /etc/kernel/postinst.d/apt-auto-removal
APT::NeverAutoRemove
{
   "^linux-image-4\.4\.0-174-generic$";
   "^linux-headers-4\.4\.0-174-generic$";
   "^linux-image-extra-4\.4\.0-174-generic$";
   "^linux-modules-4\.4\.0-174-generic$";
   "^linux-modules-extra-4\.4\.0-174-generic$";
   "^linux-signed-image-4\.4\.0-174-generic$";
   "^kfreebsd-image-4\.4\.0-174-generic$";
   "^kfreebsd-headers-4\.4\.0-174-generic$";
   "^gnumach-image-4\.4\.0-174-generic$";
   "^.*-modules-4\.4\.0-174-generic$";
   "^.*-kernel-4\.4\.0-174-generic$";
   "^linux-backports-modules-.*-4\.4\.0-174-generic$";
   "^linux-modules-.*-4\.4\.0-174-generic$";
   "^linux-tools-4\.4\.0-174-generic$";
   "^linux-cloud-tools-4\.4\.0-174-generic$";
};
/* Debug information:
# dpkg list:
# list of installed kernel packages:
```

```
# list of different kernel versions:

# Installing kernel:  ()
# Running kernel: ignored (4.4.0-174-generic)
# Last kernel:
# Previous kernel:
# Kernel versions list to keep:

# Kernel packages (version part) to protect:
4\.4\.0-174-generic
*/
### 70debconf
// Pre-configure all packages with debconf before they are installed.
// If you don't like it, comment it out.
DPkg::Pre-Install-Pkgs {"/usr/sbin/dpkg-preconfigure --apt || true";};
### docker-autoremove-suggests
Apt::AutoRemove::SuggestsImportant "false";
### docker-clean
DPkg::Post-Invoke { "rm -f /var/cache/apt/archives/*.deb /var/cache/apt/archives/partial/*.deb /var/cache/apt/*.bin || true"; };
APT::Update::Post-Invoke { "rm -f /var/cache/apt/archives/*.deb /var/cache/apt/archives/partial/*.deb /var/cache/apt/*.bin || true"; };
Dir::Cache::pkgcache ""; Dir::Cache::srcpkgcache "";
### docker-gzip-indexes
Acquire::GzipIndexes "true"; Acquire::CompressionTypes::Order:: "gz";
### docker-no-languages
Acquire::Languages "none";
```

Packages consists of a Package.gz file made with:

apt-get download <packages...> they wind up in the directory /var/cache/apt/archives

dpkg-scanpackages . /dev/null | gzip -9c > Packages.gz

/etc/apt/apt.conf.d

L The Anaconda Part

Anaconda downloads and retains the individual packages in /opt/conda/pkgs

A run of the Dockerfile, with a log of the results was made and retained.

```
docker pull continuumio/anaconda3
docker exec -it continuumio/anaconda3 /bin/bash
conda update
apt-get install -y vim
# set up for 32-bit libraries, update aptitude
dpkg --add-architecture i386
apt-get update
apt-get install -y libstdc++6:i386

apt-get install -y xorg  # add in the X11 stuff.
QUESTION ABOUT KEYBOARD!!!
apt-get install -y --reinstall procps
apt-get install -y net-tools             # ifconfig
apt install -y iproute2                  # ip in leiu of ifconfig
apt-get install -y rsyslog               # some decent logging
apt-get install -y openssh-server        # ssh into the container


...
# in the HOST window!
docker ps -a                                # get xxxxxxxx id for container.
docker stop xxxxxxxx
docker commit xxxxxxxx continuumio/c3updated # save investment at this point.
docker exec -it continuumio/c3updated /bin/bash


# Back to the new container

conda config --add channels http://ssb.stsci.edu/astroconda
conda create -n iraf27 python=2.7 iraf-all pyraf-all stsci
conda install -c conda-forge jupyter_contrib_nbextensions
useradd -m -d /home/bob -G dialout -p "$(openssl passwd -1 bobiraf)" bob
```

Things to ’cp’ to the directory iraf.aliases

fitsverify

M The Docker Part

The container's user is 'bob' with home dir as expected in /home/bob.

```
docker run -it -P --name docker_iraf \
 -v ~/Astro:/astro \
 -e DISPLAY=$DISPLAY \
 -d continuumio/c3updated

docker ps -a
# find the docker_iraf
docker commit xxxxxxxx docker_nextone
```

M.1 Dockerfile

Docker containers are built one-upon-the-other. Here we start with

continuumio/anaconda3, and command the container to load itself. We add things from our development area and eventually wind up with a working image.

M.2 Container "iraf/v0.95" the Foundation

The initial step is to create a container with the name of iraf/v0.95. This contains a few extensions to the CMachine's utilities like ps, vim, ifconfig, x11-apps (no x-11 yet), systemd, the locate utility, and creates a user called bob, with a password of 'bobiraf'. It copies the PMachine's /etc/default/keyboard file to the same place inside the container.

The next step is to get X11 running. A real chore.

```
1 ##############################################################################
  # docker build -t iraf/v0.99 . | tee > log1 2>&1
  # This is a rather heavy build of Ubuntu, feathering the nest
  # for in-container hacking. This scirpt should be field-stripped
  # and cleaned for deployment.
6 #
  # The idea here is to build a heavy container at the version=0.95
```

```
# This avoids the real heavy conda additions for IRAF.
# Having a bit of bother with the X11 crap.
#
# I suspect that a ssh -Y into the container will be in order,
# so the X in PyRAF will display in remote (the actual HOST) display
# manager.
#
# 2020-04-07T22:39:39-0600 wlg
################################################################################
FROM continuumio/anaconda3

COPY /etc/default/keyboard /etc/default/keyboard

RUN apt update && \
apt-get install -y vim && \
dpkg --add-architecture i386 && \
apt-get update && \
apt-get install -y libstdc++6:i386 && \
apt-get install -y --reinstall procps && \
apt-get install -y net-tools && \
apt-get install -y iproute2 && \
apt-get install -y rsyslog && \
apt-get install -y net-tools && \
apt-get install -y x11-apps && \
apt-get install -y systemd && \
apt-get install -y locate
RUN useradd -m -d /home/john --shell /bin/bash -G dialout -p "$(openssl passwd -1 johniraf)" john


################################################################################
# https://wiki.debian.org/DebianInstaller/Preseed#Preseeding_d-i
# pre-seed the answers...
# apt-get install -y xorg
# Notes on the above RUN targets
# apt-get install -y net-tools              # ifconfig
# apt install -y iproute2                   # ip in leiu of ifconfig
# apt-get install -y rsyslog                # some decent logging
# apt-get install -y locate                 # add locate
#conda config --add channels http://ssb.stsci.edu/astroconda
#conda create -n iraf27 python=2.7 iraf-all pyraf-all stsci
```

```
#conda install -c conda-forge jupyter_contrib_nbextensions
#
################################################################################
```

N Win10 and Docker

Docker is designed to work on native Linux boxes, and indeed shares part of the operating system. This makes Docker a very-light-weight VM-like shell. The Microsoft Windows-10 environment supports their Hyper-V (lighter than VBox but a VM non-the-less) and their WSL (Windows Sub-system for Linux). Win10 has offered other Linux-like features like their PowerShell, etc.

As of Saturday 1st January, 2022, the WSL-2 for Windows is not generally distributed. Build build 18917 or higher should support the feature, you may have to join the insider development program.

https://code.visualstudio.com/blogs/2020/03/02/docker-in-wsl2

O Docker Networking

Here the "Network Path" is considered to be the wire and packets that lead from the farthest router to where your organization meets the internet, the nearest router where your PMachine meets your organization, (the bits inside that router), where the wire enters the PMachine (this may be none or more RJ11 or wireless points), the IP stack, the pathway through the kernel, the user-space layer, and the application layer.

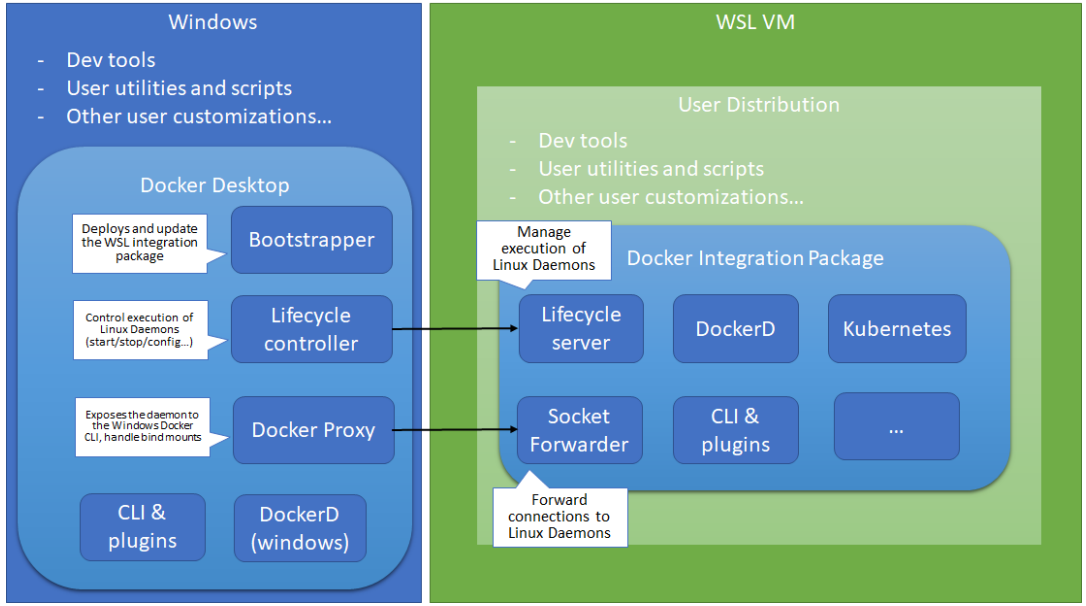


Figure N-3: Visualstudio.com redirects to Microsoft's site.Microsoft.com

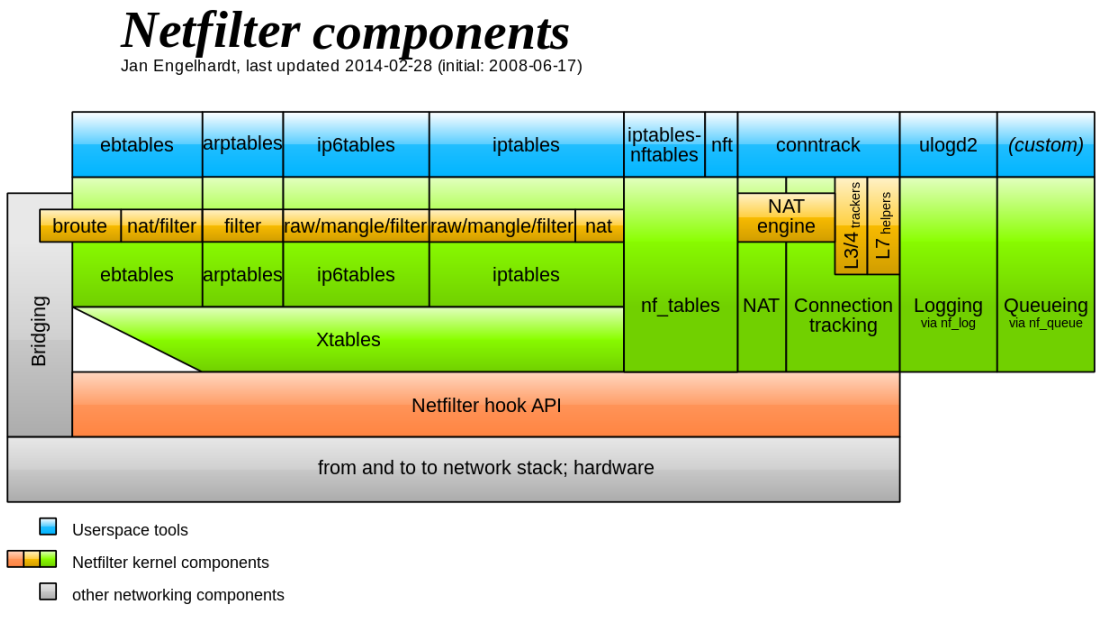


Figure O-4: Linux network layers in general. [7]

This builds on the 5-layer TCP/IP model, in that the electro/mechanical way the signal arrives inside the hardware of the PMachine can be through some 'interface'[4]

ifconfig command	New ip command
arp -a	ip neigh
arp -v	ip -s neigh
arp -s 192.168.1.1 1:2:3:4:5:6	ip neigh add 192.168.1.1 lladdr 1:2:3:4:5:6 dev eth1
arp -i eth1 -d 192.168.1.1	ip neigh del 192.168.1.1 dev eth1
ifconfig -a	ip addr
ifconfig eth0 down	ip link set eth0 down
ifconfig eth0 up	ip link set eth0 up
ifconfig eth0 192.168.1.1	ip addr add 192.168.1.1/24 dev eth0
ifconfig eth0 netmask 255.255.255.0	ip addr add 192.168.1.1/24 dev eth0
ifconfig eth0 mtu 9000 i	p link set eth0 mtu 9000
ifconfig eth0:0 192.168.1.2	ip addr add 192.168.1.2/24 dev eth0
netstat	ss
netstat -neopa	ss -neopa
netstat -g	ip maddr
route	ip route

Table 1: A ifconfig vs ip command quick summary.

O.1 IPTABLES

The Docker Container interfaces with the internet via its own bridge. The container has its own Linux layer with 'cgroups' and 'namespaces'. The Ethernet WAN, passes through iptable filtering within the main router, any intervening router, the PMachine's OS, the docker daemon and finally the container's OS. There may be a few extra layers, policies and general strategies (NAT,SNAT,DNAT,PAT) that for the rough sea for you app's packets to interoperate with the app you need at the other end.

There are a few tables:

```
# as root
2 echo "# myiptables" > /tmp/myiptables     # start the file
for f in filter nat mangle raw; do
```

[4]Run ifconfig or 'ip -addr'.

```
  echo "# table type : $f" >> /tmp/myiptables
  iptables -t $f -L >> /tmp/myiptables
done
```

The output:

```
# myiptables
# table type :filter
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy DROP)
target     prot opt source               destination
DOCKER-USER  all  --  anywhere             anywhere
DOCKER-ISOLATION-STAGE-1  all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere             ctstate RELATED,ESTABLISHED
DOCKER     all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain DOCKER (1 references)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             172.17.0.4           tcp dpt:postgresql
ACCEPT     tcp  --  anywhere             172.17.0.3           tcp dpt:postgresql

Chain DOCKER-ISOLATION-STAGE-1 (1 references)
target     prot opt source               destination
DOCKER-ISOLATION-STAGE-2  all  --  anywhere             anywhere
RETURN     all  --  anywhere             anywhere

Chain DOCKER-ISOLATION-STAGE-2 (1 references)
target     prot opt source               destination
DROP       all  --  anywhere             anywhere
RETURN     all  --  anywhere             anywhere

Chain DOCKER-USER (1 references)
target     prot opt source               destination
RETURN     all  --  anywhere             anywhere
# table type :nat
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
DOCKER     all  --  anywhere             anywhere             ADDRTYPE match dst-type LOCAL

Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
DOCKER     all  --  anywhere            !localhost/8          ADDRTYPE match dst-type LOCAL

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
MASQUERADE  all  --  172.17.0.0/16        anywhere
MASQUERADE  tcp  --  172.17.0.4           172.17.0.4           tcp dpt:postgresql
MASQUERADE  tcp  --  172.17.0.3           172.17.0.3           tcp dpt:postgresql

Chain DOCKER (2 references)
target     prot opt source               destination
RETURN     all  --  anywhere             anywhere
DNAT       tcp  --  anywhere             anywhere             tcp dpt:55432 to:172.17.0.4:5432
DNAT       tcp  --  anywhere             anywhere             tcp dpt:55435 to:172.17.0.3:5432
# table type :mangle
Chain PREROUTING (policy ACCEPT)
```

```
target     prot opt source               destination

Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
# table type :raw
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

In General IP Tables enter the picture.

https://www.cyberciti.biz/tips/linux-iptables-examples.html

Has a decent description of commands etc.

Change commands require root privileges This includes query commands and commands that cause changes. Use sudo for each command, or sudo -s (dangerous way) and have at it.

In particular the

```
iptables -L --line-numbers   # See the rules with numbers
iptables -D INPUT 3          # Find the particular one, -D delete it from IMPUT side.
```

P Cleaning House

This little ditty removes ALL the images.

```
sudo docker rmi $(sudo docker images -aq) --force
```

Q IRAF/PyRAF Multiverse

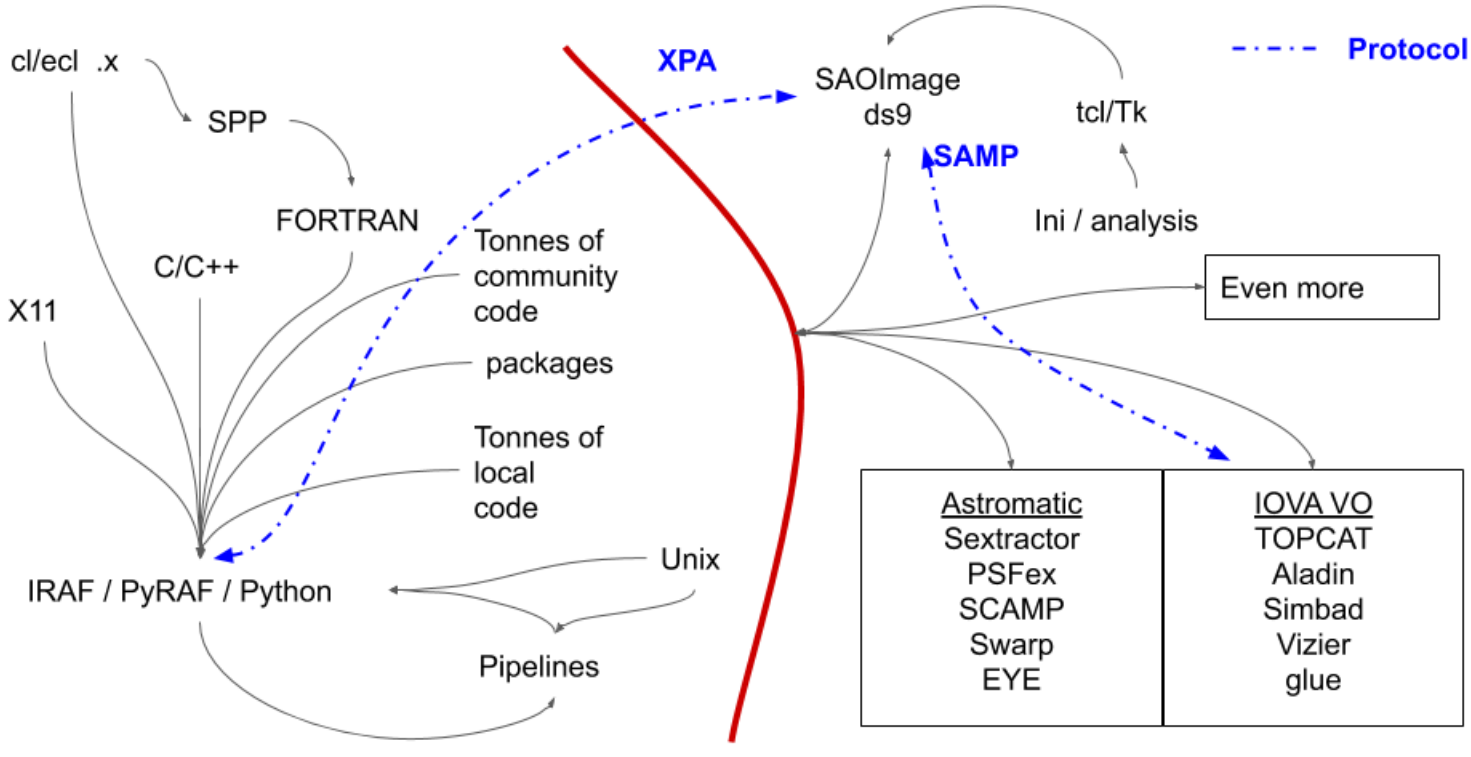


Figure Q-5: IRAF/PyRAF Multiverse of players.

IRAF[18] [5], eventually grew into 'cl' or command language. IRAF is a 'collection' of routines, from various authors at various institutions, designed to work with the institutions data. Eventually, the community settled on NOAO coordinating both IRAF, its language mechanisms and distribution; and the rough agreement for image transfer using FITS[4] (Flexible Image Transport System).Remember, the first word is "Flexible".

Its complicated: See Figure Q-5.

IRAF's basic scripting language (cl) was quickly extended into the modern ecl (extended-cl). Rather than an "eecl", Python's readline command line interpreter [9, 6]) was adapted to create PyRAF[8] – to support most existing ecl 'tasks' together with their module subroutine equivalents. However, copyrighted algorithms and a heavy dependency on early X11 has IRAF 'wedged' [7].

While PyRAF/IRAF has greatly reduced its dependency on i386 libraries, it still heavily depends on early X-Windows code[6]. Ubuntu, a popular Linux operating system, is dropping 32-bit (i386) program support with the release of 19.04 (2019). The X11 replacement "Wayland" and other attempts to replace X11 have met with limited successes. X11 support was phased out of the Apple OSs since the 2012 framework and has relied on external packages like XQuartz etc. PyRAF/IRAF has resolved licensing issues from use of algorithms from "Numerical Recipes for C" [12] - a popular academic reference in the day.

PyRAF/IRAF is under pressure from the younger astronomers putting a heavy reliance of Python, Astropy, and very modern computing environments. This is slow-going as the inspired receive their PhDs and their efforts wane in the face of new duties.

This adds up to the fact that a very large user base (by astronomy scales) is losing the ability to perform analysis and/or suffer from a daunting learning AND development curve!

These notes explore bridging the PyRAF/IRAF and Astropy communities. This article focuses on moving into a Ubuntu 18.04+ OS image within a Docker Container that will be available on Linux/Mac/Win10 platforms.

The one other issue to put into people's brains is Anaconda's reliance on the Intel Math Kernel Library for Numpy is broken on AMD processors.

Carl Boettiger [2] makes a good argument for disseminating scientific projects in a 'ready to run' docker based environment.

[5] IRAF (Image Reduction (and) Analysis Facility)

[6] X has essentially been frozen at X11 since 1986. The font generation is downright primitive.

R AAVSO

Enter star name; browser, save the image; use photometry like cut/paste: the paste is more of a tsv but with issues:

1. Lable is 'Chart ID'
2. There is no link to get a decent TSV/CSV/XLS.
3. There are embedded UNICODE eyecandy characters.
4. Some columns have two fields! STUPID.
   (a) RA and DEC
   (b) V has goofy mag statement X.XXX (X.XXX)XX 3 fields?
   (c) B-V is a DELTA with (X.XXX)
5. 000-BKP-763 is cited with missing mag error, where B-V has an error.
6. 000-BKP-763 has missing mag error as a unicode 0x2014.
7. Recommend nan, can not error-propagate nan.
8. Make a data dictionary/algorithm dictionary.

Recommend RA/DEC be RA_ICRS/DEC_ICRS and be in degrees no decorator (put the decorator as [deg] in header RA_2000/DEC_2000 be in sexigesimal hours Magnitudes be split into 2 or 3 columns Footnote to table explaining fields. Make fields machine parsable with python snippets to parse.

S Planning

```
Observations
  usw
     targets
        targetname
          <stuff>
```

S.1 ds9

General: Set the font sizes: Default is 10, may want 12 Bllank/inf/nan to yellow (actually get a chance to see one) Startup: Initialize XPA and offer SAMP connection (Tie to TOPCAT and Aladin)

Menus and Buttons : See below.

Magnifier: 8x or 16x really zoom into a 'star'

Edit $\rightarrow$ preferences.

Graphs: Axis linear; Method Average/sum Annulus!

Analysis log>?

Pixel Table: 13 (bigger is better) Catalogs : Color Cyan

Postscript: DPI 600 (for publication)

S.1.1 Menus and Buttons

Here certain preferences are set. Very Important: first off, use retions and projection. This makes the projection tool the default one.

You can adjust what is ticked as default in the menus and what appears in the button bar. For example: wake up in vertical mode.

S.1.2 Finder Charts

Load the image. Under Analysis $\rightarrow$ coordinate and coordinate grid:

In the menu; you may save the file. Save this to the general usw dir for the instrument.

```
Type
Publication
Exterior Axes
Exterior Numerics

Coordinate
WCS
ICRS
degrees or (Sexigesimal)

Grid
Color Black (or White)
```

```
Line (thickness or dash)

Axes
Color Black

Numerics
Color black
Font (size)

Labels
Color
Font

Tickmakrs
Color
Line (size)

Border (It sometimes wraps!)
Color Black
Line (2 a tad thicker)

Labels:
Axis 1 RA (IRCS [deg]) uncheck the box
Axis 2 Dec (IRCS [deg]) uncheck the box

Spacing:
Title   12
Axis 1   0
Axis 2   5
```

S.1.3 Hacking ds9

Only to say, you can add your own tcl to ds9.ini. That is very powerful.

```
Analysis →
  Survey →  POSS2/UKSTU (or other red)
  Use 45 arcmin as the FOV
  Save file <target>_DSS_Finder
Analysis →  Catalogs →  Database  →  SIMBAD
```

```
Catalog Tool:  →  Symbol Color  →  Cyan
Catalog Tool:  →  Symbol Advanced
   Text $FLUX_B
```

T Handy Docker Phrases

```
docker run -it --hostname myfoo # override the default ID as hostname.
```

Bibliography and References

[1] E. Bertin et.al. Sextractor v2.13 draft. *WEB/URL*, 2015. https://www.astromatic.net/pubsvn/software/sextractor/trunk/doc/sextractor.pdf.

[2] Carl Boettiger. An introduction to docker for reproducible research, with examples from the R environment. *CoRR*, abs/1410.0846, 2014.

[3] F. Bonnarel, P. Fernique, O. Bienaymé, D. Egret, F. Genova, M. Louys, F. Ochsenbein, M. Wenger, and J. G. Bartlett. The ALADIN interactive sky atlas. A reference tool for identification of astronomical sources. *Astronomy and Astrophysics Supplement Series*, 143:33–40, April 2000.

[4] et. al. Chiappetti, L. Fits standard document version 4.0 [draft] (22 july 2016). *WEB/URL*, 2016. https://fits.gsfc.nasa.gov/fits_standard.html.

[5] S. G. Djorgovski and C. Beichman. New Digital Sky Surveys: Introductory Comments. In *American Astronomical Society Meeting Abstracts #192*, volume 192 of *American Astronomical Society Meeting Abstracts*, page 64.01, May 1998.

[6] C. E. Downey, Douglas Tody, and George H. Jacoby. Cl programmers guide. *WEB/URL*, 1982. ftp://iraf.noao.edu/ftp/docs/clman.ps.Z.

[7] Jan. Engelhardt. Relation of the different netfilter components to another. *WEB/URL*, 2008. https://en.wikipedia.org/wiki/Netfilter#/media/File:Netfilter-components.svg.

[8] P. Greenfield and R. L. White. Where Will PyRAF Lead Us? The Future of Data Analysis Software at STScI. In A. M. Koekemoer, P. Goudfrooij, and L. L. Dressel, editors, *The 2005 HST Calibration Workshop: Hubble After the Transition to Two-Gyro Mode*, page 437, January 2006.

[9] Philip E. Hodge. Pyraf programmer's guide. *WEB/URL*, 2004. http://stsdas.stsci.edu/stsci_python_sphinxdocs_2.13/docs/pyraf_guide.pdf https://lancesimms.com/programs/Python/pyraf/Docs/pyraf_guide.pdf.

[10] W. A. Joye and E. Mandel. *New Features of SAOImage DS9*, volume 295 of *Astronomical Society of the Pacific Conference Series*, page 489. 2003.

[11] D. Lang, D. W. Hogg, K. Mierle, M. Blanton, and S. Roweis. Astrometry.net: Blind astrometric calibration of arbitrary astronomical images. *The Astronomical Journal*, 137:1782–2800, 2010. arXiv:0910.2233,doi:10.1088/0004-6256/139/5/1782](http://dx.doi.org/10.1088/0004-6256/139/5/1782).

[12] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C*. Cambridge University Press, Cambridge, USA, second edition, 1992.

[13] A. M. Price-Whelan, B. M. Sipőcz, H. M. Günther, P. L. Lim, S. M. Crawford, S. Conseil, D. L. Shupe, M. W. Craig, N. Dencheva, A. Ginsburg, J. T. VanderPlas, L. D. Bradley, D. Pérez-Suárez, M. de Val-Borro, (Primary Paper Contributors, T. L. Aldcroft, K. L. Cruz, T. P. Robitaille, E. J. Tollerud, (Astropy Coordination Committee, C. Ardelean, T. Babej, Y. P. Bach, M. Bachetti, A. V. Bakanov, S. P. Bamford, G. Barentsen, P. Barmby, A. Baumbach, K. L. Berry, F. Biscani, M. Boquien, K. A. Bostroem, L. G. Bouma, G. B. Brammer, E. M. Bray, H. Breytenbach, H. Buddelmeijer, D. J. Burke, G. Calderone, J. L. Cano Rodríguez, M. Cara, J. V. M. Cardoso, S. Cheedella, Y. Copin, L. Corrales, D. Crichton, D. D'Avella, C. Deil, É. Depagne, J. P. Dietrich, A. Donath, M. Droettboom, N. Earl, T. Erben, S. Fabbro, L. A. Ferreira, T. Finethy, R. T. Fox, L. H. Garrison, S. L. J. Gibbons, D. A. Goldstein, R. Gommers, J. P. Greco, P. Greenfield, A. M. Groener, F. Grollier, A. Hagen, P. Hirst, D. Homeier, A. J. Horton, G. Hosseinzadeh, L. Hu, J. S. Hunkeler, Ž. Ivezić, A. Jain, T. Jenness, G. Kanarek, S. Kendrew, N. S. Kern, W. E. Kerzendorf, A. Khvalko, J. King, D. Kirkby, A. M. Kulkarni, A. Kumar, A. Lee, D. Lenz, S. P. Littlefair, Z. Ma, D. M. Macleod, M. Mastropietro, C. McCully, S. Montagnac, B. M. Morris, M. Mueller, S. J. Mumford, D. Muna, N. A. Murphy, S. Nelson, G. H. Nguyen, J. P. Ninan, M. Nöthe, S. Ogaz, S. Oh, J. K. Parejko, N. Parley, S. Pascual, R. Patil, A. A. Patil, A. L. Plunkett, J. X. Prochaska, T. Rastogi, V. Reddy Janga, J. Sabater, P. Sakurikar, M. Seifert, L. E. Sherbert, H. Sherwood-Taylor, A. Y. Shih, J. Sick, M. T. Silbiger, S. Singanamalla, L. P. Singer, P. H. Sladen, K. A. Sooley, S. Sornarajah, O. Streicher, P. Teuben,

S. W. Thomas, G. R. Tremblay, J. E. H. Turner, V. Terrón, M. H. van Kerkwijk, A. de la Vega, L. L. Watkins, B. A. Weaver, J. B. Whitmore, J. Woillez, V. Zabalza, and (Astropy Contributors. The Astropy Project: Building an Open-science Project and Status of the v2.0 Core Package. *The Astronomical Journal*, 156:123, September 2018.

[14] M. Taylor, T. Boch, M. Fitzpatrick, A. Allan, J. Fay, L. Paioro, J. Taylor, and D. Tody. Simple Application Messaging Protocol Version 1.3. IVOA Recommendation 11 April 2012, April 2012.

[15] M. Taylor, T. Boch, M. Fitzpatrick, A. Allan, J. Fay, L. Paioro, J. Taylor, and D. Tody. Simple application messaging protocol version 1.3. *WEB/URL*, 2013.
http://www.ivoa.net/documents/SAMP/20120411/REC-SAMP-1.3-20120411.html
local: SAMP_SimpleApplicationMessagingProtocol.pdf.

[16] M. B. Taylor. *TOPCAT & STIL: Starlink Table/VOTable Processing Software*, volume 347 of *Astronomical Society of the Pacific Conference Series*, page 29. 2005.

[17] M. B. Taylor. *TOPCAT's TAP Client*, volume 512 of *Astronomical Society of the Pacific Conference Series*, page 589. 2017.

[18] F. Valdes. The Interactive Data Reduction and Analysis Facility (IRAF). In *Bulletin of the American Astronomical Society*, volume 16 of *Bull. Am. Astron. Soc.*, page 497, March 1984.

[19] P. Škoda, P. W. Draper, M. C. Neves, D. Andrešič, and T. Jenness. Spectroscopic analysis in the virtual observatory environment with SPLAT-VO. *Astronomy and Computing*, 7:108–120, November 2014.

Action Items/EndNotes:

[1]ACTION: The origin of a 'hypervisor' is hard to nail down but today it is usually interpreted to mean running one OS within another.

[2]ACTION: The container is built using the continuum/anaconda container using a reasonably current core Ubuntu 18.04 OS. The "Dockerfile" for the container is in Appendix M.1. Remember, Ubuntu 18.04 still supports 32-bit libraries.

[3]ACTION: Any mass of things needed by one or more contaienrs.

[4]ACTION: Get a crib together for this.

[5]ACTION: Add a way to make verbatim figures stand out.

[6]ACTION: An excellent way to grab a package and see what it brings along with it without changing PMachine host.

[7]***wedged adj.***

1. To be stuck, incapable of proceeding without help. This is different from having crashed. If the system has crashed, it has become totally non-functioning. If the system is wedged, it is trying to do something but cannot make progress; it may be capable of doing a few things, but not be fully operational. For example, a process may become wedged if it deadlocks with another (but not all instances of wedging are deadlocks). See also gronk, locked up, hosed, hung (wedged is more severe than hung). 2. Often refers to humans suffering misconceptions. "He's totally wedged – he's convinced that he can levitate through meditation." 3. [Unix] Specifically used to describe the state of a TTY left in a losing state by abort of a screen-oriented program or one that has messed with the line discipline in some obscure way.

There is some dispute over the origin of this term. It is usually thought to derive from a common description of recto-cranial inversion; however, it may actually have originated with older 'hot-press' printing technology in which physical type elements were locked into type frames with wedges driven in by mallets. Once this had been done, no changes in the typesetting for that page could be made.

For this and more jargon: https://www.eps.mcgill.ca/jargon/jargon.#wedged

Index