Introduction - cf-python and cf-plot

The "cf" in cf-python and cf-plot are to indicate that they are underpined by CF (Climate and Forecast) Metadata Conventions:

[http://cfconventions.org (http://cfconventions.org)](http://cfconventions.org)

---

cf-python - The python cf package implements the CF data model for the reading, writing and processing of data and metadata:

[https://ncas-cms.github.io/cf-python (https://ncas-cms.github.io/cf-python)](https://ncas-cms.github.io/cf-python)

---

cf-plot - A set of Python routines for making the common contour, vector and line plots that climate researchers use. can also plot Numpy arrays of data:

[http://ajheaps.github.io/cf-plot (http://ajheaps.github.io/cf-plot)](http://ajheaps.github.io/cf-plot)

Read, select, write example

In [1]:

```
# Inline images in Ipython Notebook - not needed in a terminal Python session.
%matplotlib inline
```

In [2]:

```
# Import cf-python and cf-plot
import cf
import cfplot as cfp
```

In [3]:

```
# Read a data file
fl = cf.read('ncas_data/data1.nc')
```

In [4]:

```
# View the contents of the file
fl
```

Out[4]:

```
[<CF Field: divergence_of_wind(time(1), pressure(23), latitude(160), longitude(320)) s**-1>,
 <CF Field: long_name=Ozone mass mixing ratio(time(1), pressure(23), latitude(160), longitude(3
20)) kg kg**-1>,
 <CF Field: long_name=Potential vorticity(time(1), pressure(23), latitude(160), longitude(320))
 K m**2 kg**-1 s**-1>,
 <CF Field: specific_humidity(time(1), pressure(23), latitude(160), longitude(320)) kg kg**-1>,
 <CF Field: relative_humidity(time(1), pressure(23), latitude(160), longitude(320)) %>,
 <CF Field: atmosphere_relative_vorticity(time(1), pressure(23), latitude(160), longitude(320))
 m**2 s**-1>,
 <CF Field: air_temperature(time(1), pressure(23), latitude(160), longitude(320)) K>,
 <CF Field: eastward_wind(time(1), pressure(23), latitude(160), longitude(320)) m s**-1>,
 <CF Field: northward_wind(time(1), pressure(23), latitude(160), longitude(320)) m s**-1>,
 <CF Field: atmosphere_relative_vorticity(time(1), pressure(23), latitude(160), longitude(320))
 s**-1>,
 <CF Field: divergence_of_wind(time(1), pressure(23), latitude(160), longitude(320)) m**2 s**-1
>,
 <CF Field: vertical_air_velocity_expressed_as_tendency_of_pressure(time(1), pressure(23), lati
tude(160), longitude(320)) Pa s**-1>,
 <CF Field: geopotential(time(1), pressure(23), latitude(160), longitude(320)) m**2 s**-2>]
```

In [4]:

```
# Select the air temperature
temp = fl.select('air_temperature')[0]
temp
```

Out[4]:

```
<CF Field: air_temperature(time(1), pressure(23), latitude(160), longitude(320)) K>
```

In [5]:

```
# Select by index
temp = fl[6]
temp
```

Out[5]:

```
<CF Field: air_temperature(time(1), pressure(23), latitude(160), longitude(320)) K>
```

In [6]:

```
print(temp)
```

```
Field: air_temperature (ncvar%T)
--------------------------------
Data            : air_temperature(time(1), pressure(23), latitude(160), longitude(320)) K
Dimension coords: time(1) = [1964-01-21 00:00:00]
                : pressure(23) = [1000.0, ..., 1.0] mbar
                : latitude(160) = [89.14151763916016, ..., -89.14151763916016] degrees_north
                : longitude(320) = [0.0, ..., 358.875] degrees_east
```

In [7]:

```
# Select by long_name
vorticity = fl.select('long_name=Potential vorticity')[0]
```

In [8]:

```
# See a longer list of field contents
print(vorticity)
```

```
Field: long_name=Potential vorticity (ncvar%PV)
-----------------------------------------------
Data            : long_name=Potential vorticity(time(1), pressure(23), latitude(160), longitude
(320)) K m**2 kg**-1 s**-1
Dimension coords: time(1) = [1964-01-21 00:00:00]
                : pressure(23) = [1000.0, ..., 1.0] mbar
                : latitude(160) = [89.14151763916016, ..., -89.14151763916016] degrees_north
                : longitude(320) = [0.0, ..., 358.875] degrees_east
```

In [9]:

```
# Set the standard_name of the field
vorticity.standard_name = 'ertel_potential_vorticity'

# Look at field contents
print(vorticity)
```

```
Field: ertel_potential_vorticity (ncvar%PV)
-------------------------------------------
Data            : ertel_potential_vorticity(time(1), pressure(23), latitude(160), longitude(320
)) K m**2 kg**-1 s**-1
Dimension coords: time(1) = [1964-01-21 00:00:00]
                : pressure(23) = [1000.0, ..., 1.0] mbar
                : latitude(160) = [89.14151763916016, ..., -89.14151763916016] degrees_north
                : longitude(320) = [0.0, ..., 358.875] degrees_east
```

In [10]:

```
# Write the modified field to a netCDF file
cf.write(vorticity, 'newfile.nc')
```

Contour plots

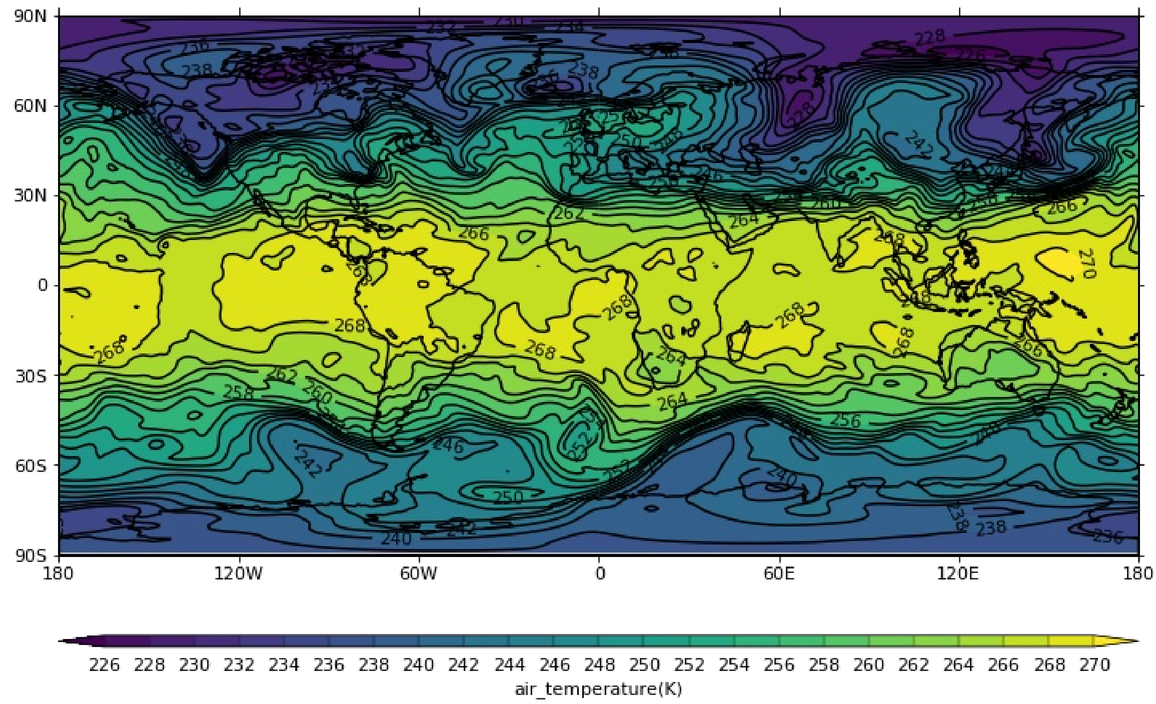
In [11]:

```
# Use subspace to select the temperature at 500mb
t_500 = temp.subspace(pressure=500)
print(t_500)
```

```
Field: air_temperature (ncvar%T)
--------------------------------
Data            : air_temperature(time(1), pressure(1), latitude(160), longitude(320)) K
Dimension coords: time(1) = [1964-01-21 00:00:00]
                : pressure(1) = [500.0] mbar
                : latitude(160) = [89.14151763916016, ..., -89.14151763916016] degrees_north
                : longitude(320) = [0.0, ..., 358.875] degrees_east
```

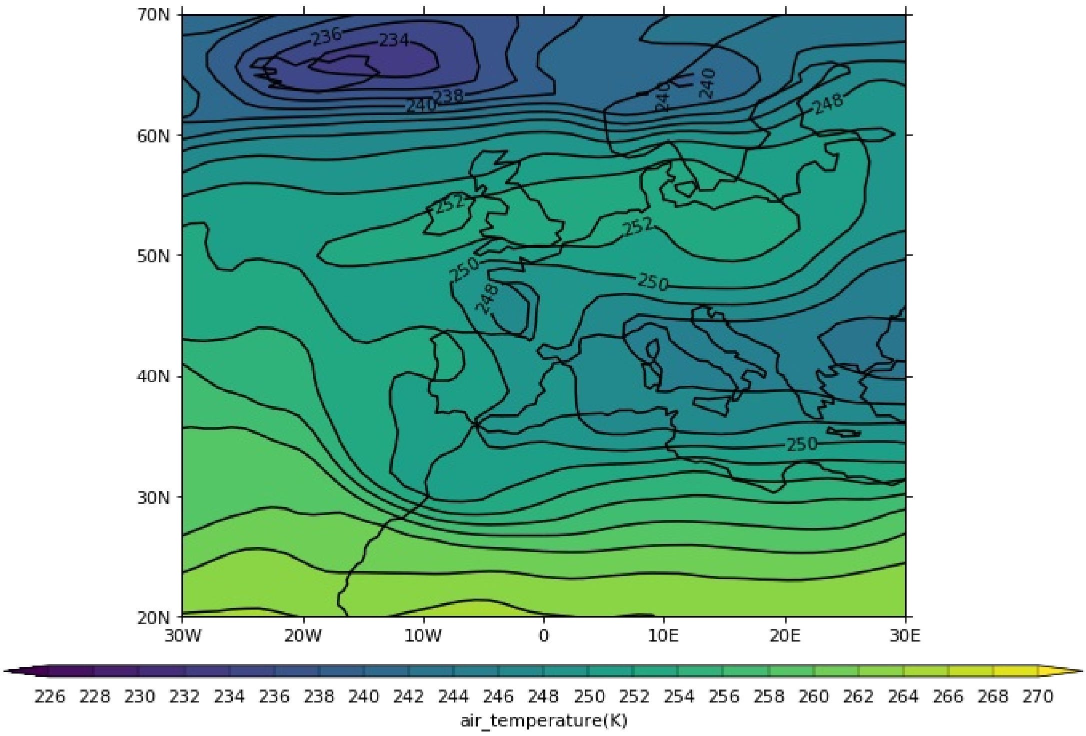
In [12]:

```
# Make a contour plot of the data
cfp.con(t_500)
```



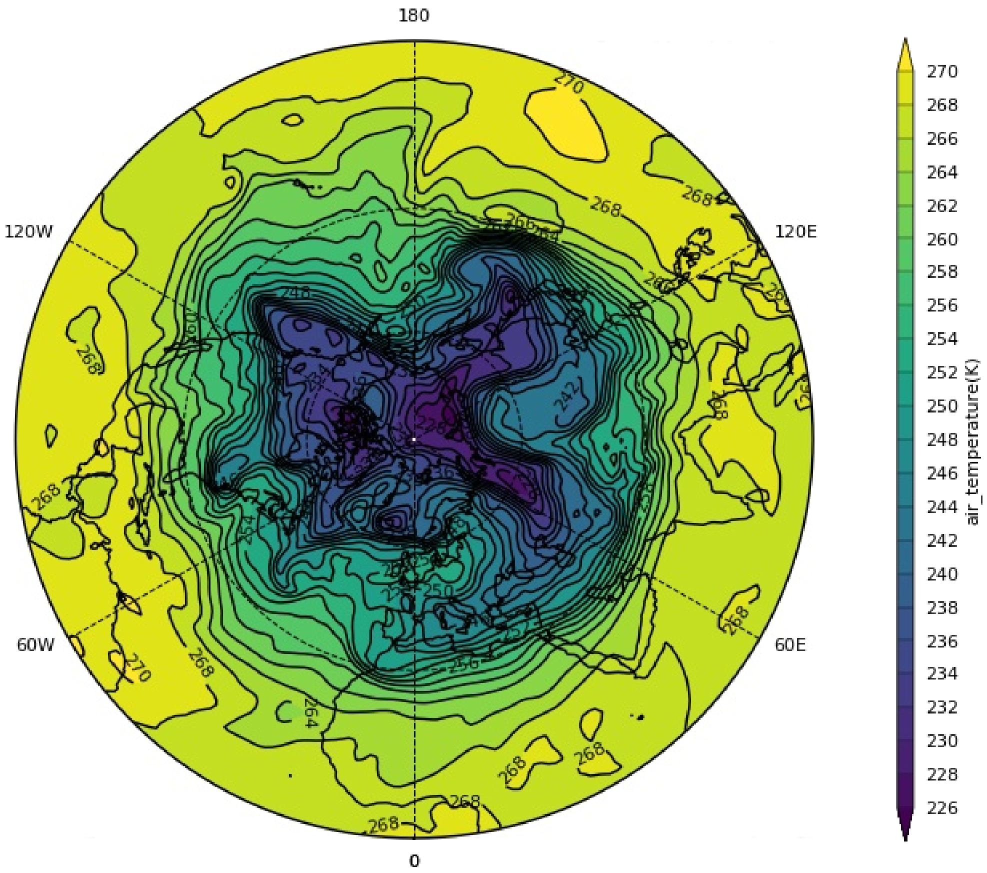
In [13]:

```
# Use mapset to select Europe and make a new contour plot
cfp.mapset(lonmin=-30, lonmax=30, latmin=20, latmax=70)
cfp.con(t_500)
```



In [14]:

```
# Make a Northern Hemiphere polar stereographic plot
cfp.mapset(proj='npstere')
cfp.con(t_500)
```



In [15]:

```
# Reset mapping
cfp.mapset()
```

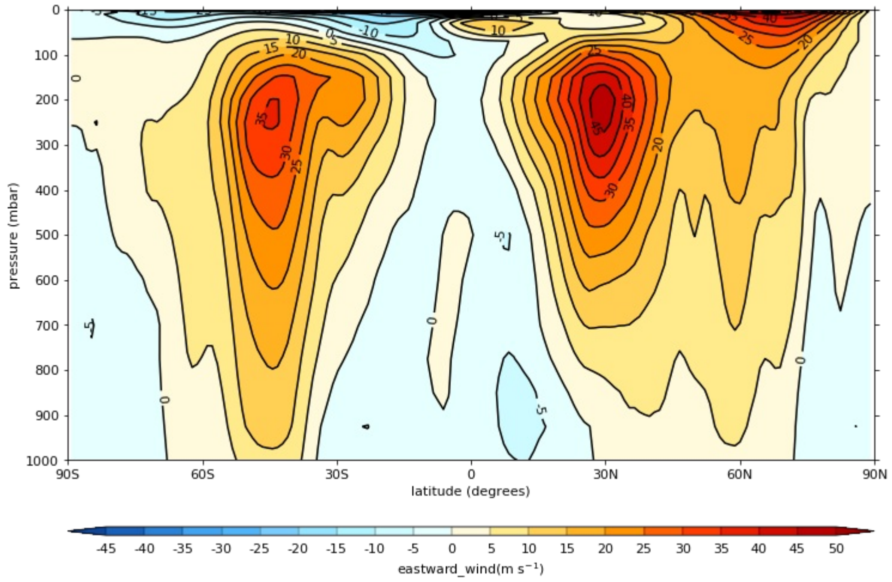
In [16]:

```
# Select the zonal wind and make a zonal mean of this using the collapse function in cf-python
u = fl.select('eastward_wind')[0]
u_mean = u.collapse('longitude: mean')
print(u_mean)
```

```
Field: eastward_wind (ncvar%U)
------------------------------
Data            : eastward_wind(time(1), pressure(23), latitude(160), longitude(1)) m s**-1
Cell methods    : longitude(1): mean
Dimension coords: time(1) = [1964-01-21 00:00:00]
                : pressure(23) = [1000.0, ..., 1.0] mbar
                : latitude(160) = [89.14151763916016, ..., -89.14151763916016] degrees_north
                : longitude(1) = [179.4375] degrees_east
```

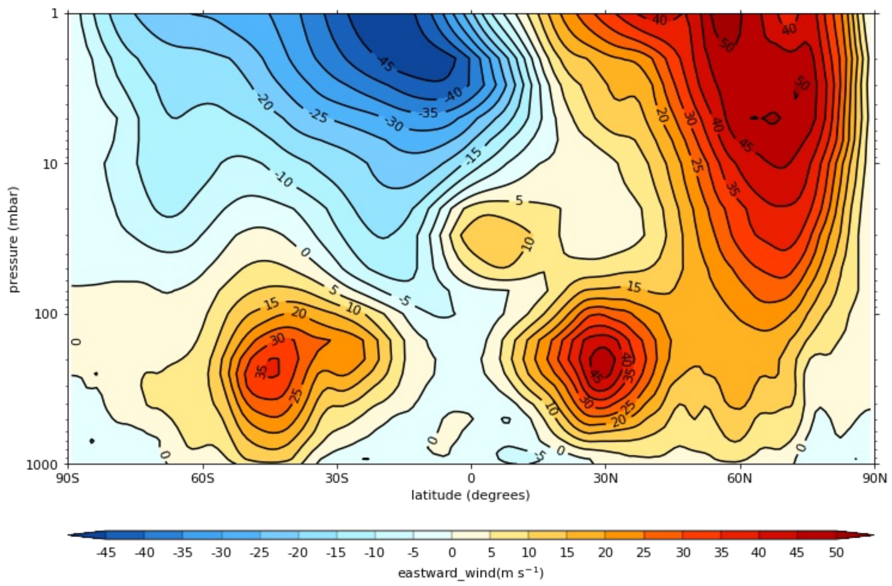
In [17]:

```
# Make a zonal mean zonal wind plot
cfp.con(u_mean)
```



In [18]:

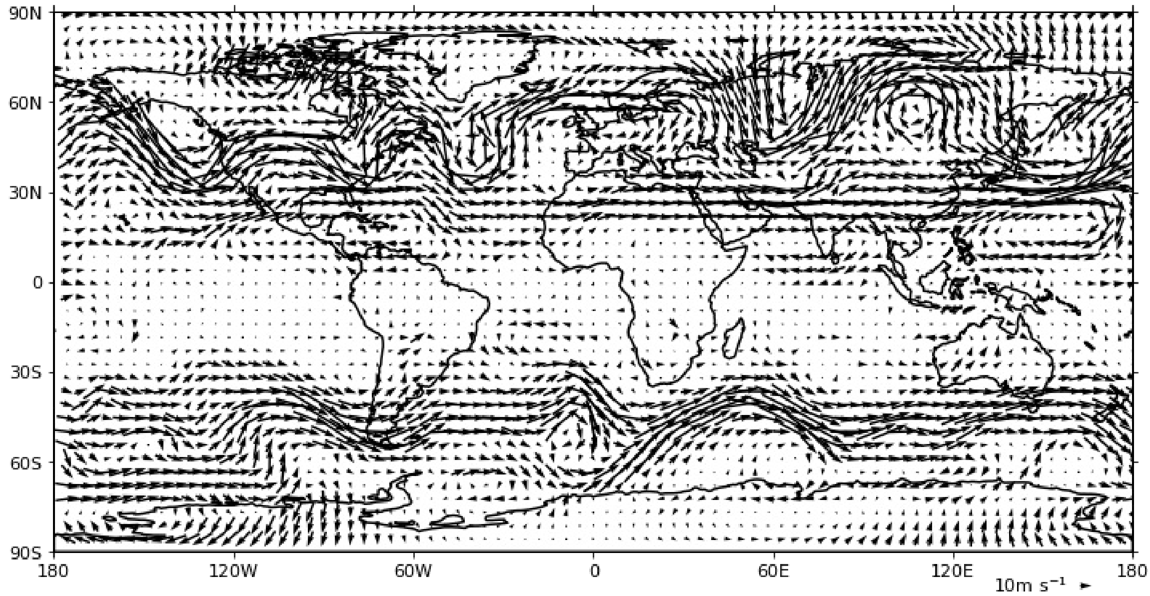
```
# Make a log y-axis plot of the zonal mean zonal wind
cfp.con(u_mean, ylog=True)
```



Vector plots

In [19]:

```
# Select u and v wind components at 500mb and make a vector plot
# We use a stride of 4 in plotting the vectors as the points are close together
u = fl.select('eastward_wind')[0].subspace(pressure=500)
v = fl.select('northward_wind')[0].subspace(pressure=500)
cfp.vect(u=u, v=v, key_length=10, scale=100, stride=4)
```



Line plots

In [20]:

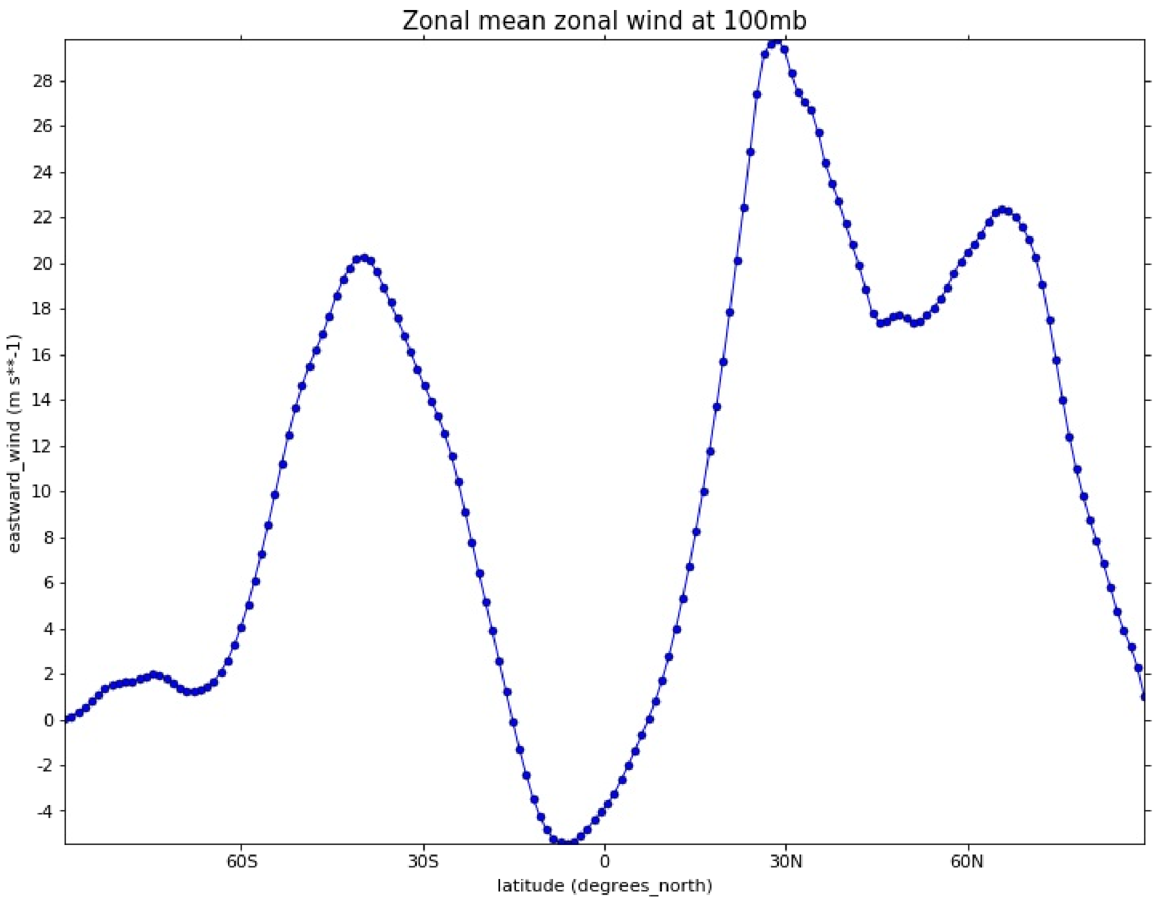
```
# Select the zonal mean zonal wind at 100mb
u = fl.select('eastward_wind')[0]
u_mean = u.collapse('longitude: mean')
u_mean_100 = u_mean.subspace(pressure=100)

print(u_mean_100)
```

```
Field: eastward_wind (ncvar%U)
------------------------------
Data            : eastward_wind(time(1), pressure(1), latitude(160), longitude(1)) m s**-1
Cell methods    : longitude(1): mean
Dimension coords: time(1) = [1964-01-21 00:00:00]
                : pressure(1) = [100.0] mbar
                : latitude(160) = [89.14151763916016, ..., -89.14151763916016] degrees_north
                : longitude(1) = [179.4375] degrees_east
```

In [21]:

```
cfp.lineplot(u_mean_100, marker='o', color='blue', title='Zonal mean zonal wind at 100mb')
```



Regridding

Regrid some temperature longitude-latitude data to another grid and make a plot of the difference between the two datsets.

In [22]:

```
# Read in data on two different grids
temp_era40 = cf.read('ncas_data/data2.nc')[0]
temp_era_in = cf.read('ncas_data/data3.nc')[0]

print(temp_era40)
print(temp_era_in)
```

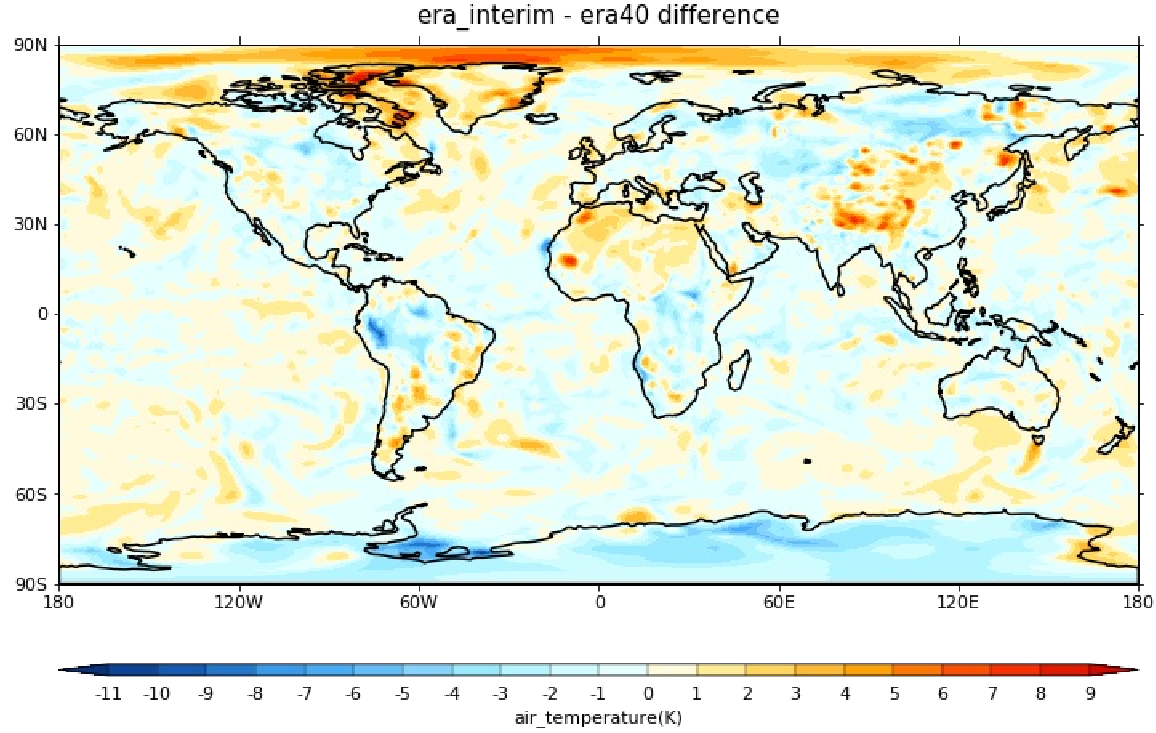
```
Field: air_temperature (ncvar%T)
--------------------------------
Data            : air_temperature(long_name=t(1), long_name=p(1), latitude(160), longitude(320)
) K
Dimension coords: long_name=t(1) = [1981-01-21 00:00:00]
                : long_name=p(1) = [1000.0] mbar
                : latitude(160) = [89.14151763916016, ..., -89.14151763916016] degrees_north
                : longitude(320) = [0.0, ..., 358.875] degrees_east
Field: air_temperature (ncvar%T)
--------------------------------
Data            : air_temperature(long_name=t(1), long_name=p(1), long_name=latitude(256), long
_name=longitude(512)) K
Dimension coords: long_name=t(1) = [1981-01-21 00:00:00]
                : long_name=p(1) = [1000.0] mbar
                : long_name=latitude(256) = [89.46294403076172, ..., -89.46294403076172] degree
s_north
                : long_name=longitude(512) = [0.0, ..., 359.296875] degrees_east
```

In [23]:

```
# Perform the regridding
temp_regrid = temp_era_in.regrids(temp_era40, method='bilinear')
```

In [24]:

```
# Make a contour plot of the difference between the two datasets
cfp.con(temp_regrid - temp_era40, lines=False, title='era_interim - era40 difference')
```



cf-plot gallery: [http://ajheaps.github.io/cf-plot/gallery.html (http://ajheaps.github.io/cf-plot/gallery.html)](http://ajheaps.github.io/cf-plot/gallery.html)

cf-python functionality: [https://ncas-cms.github.io/cf-python (https://ncas-cms.github.io/cf-python)](https://ncas-cms.github.io/cf-python)

- ### read field constructs from netCDF, PP and UM datasets,
- ### create new field constructs in memory,
- ### inspect field constructs,
- ### test whether two field constructs are the same,
- ### modify field construct metadata and data,
- ### create subspaces of field constructs,
- ### write field constructs to netCDF datasets on disk,
- ### incorporate, and create, metadata stored in external files,
- ### read, write, and create data that have been compressed by convention (i.e. ragged or gathered arrays), whilst presenting a view of the data in its uncompressed form,
- ### Combine field constructs arithmetically,
- ### Manipulate field construct data by arithmetical and trigonometrical operations,
- ### Perform statistical collapses on field constructs,
- ### Perform histogram, percentile and binning operations on field constructs,
- ### Regrid field constructs,
- ### Apply convolution filters to field constructs,
- ### Calculate derivatives of field constructs,
- ### Create field constructs to create derived quantities (such as vorticity).