**Pandas Cheat Sheet for Beginners. Compiled by Sisay Zeleke tosisay@gmail.com**

**Create Series(Single dimension labeled array holding any dtype)**

**s1 = pd.Series([‘d1’, ‘d2’, ‘d3’,…])** *#With default index starting with 0*

**s2 = pd.Series([‘d1’, ‘d2’, ‘d3’,….], index=[‘idx0’, ‘idx1’, ‘idx3’,…])** *#With custom index with any dtype (not necessarily need to be unique but equal length with data)*

**s3 = pd.Series({‘d'1’: {‘subd1’, ‘subd2’, ‘subd3’,…}, ‘d2’: {‘subd1’, ‘subd2’, ‘subd3’,….}})** *#With custom hierarchical data using dictionary (this e.g contains len of 2)*

**s4 = pd.Series({‘d'1’: [‘subd1’, ‘subd2’, ‘subd3’,…], ‘d2’: [‘subd1’, ‘subd2’, ‘subd3’,….]})**  *#Dictionary of lists*

**s1.where(s1 > 50)** *#Subset the data*

**s1[(s1 > 50)]**

**s1[~(s1 < 10) | (s1 < 20)]**

**s = df[‘colname’]** *#Select col to Series*

**S = df.pop(‘colname’)** *#Get and drop col from df*

**Create Dataframe**

*#Define data as per each column/column-wise*

**data = {'Subcity': ['Adis Ketema', 'Arada', 'Kolfe', 'Kirkos'],**

**'Downtown': ['Bole', 'Megenagna', 'Mexico', 'Piasa'],**

**'Population': [122541, 189625, 154813, 211467]**

**}**

*#Or with index,*

**data = {'Subcity': {0: 'Adis Ketema', 1: 'Arada', 2: 'Kolfe', 3: 'Kirkos'},**

**'Downtown': {0: 'Bole', 1: 'Megenagna', 2: 'Mexico', 3: 'Piasa'},**

**'Population': {0: 122541, 1: 189625, 2: 154813, 3: 211467}**

**}**

**df = pd.DataFrame(data=data, index=['a','b','b','d'], columns=['Subcity', 'Downtown', 'Population'])** *#Columns is optional since dictionary key is assumed as col heading and dictionary values are assumed as col data. If supplied col heading is different than the dictionary keys, it will be assumed as new col, I,e; ‘Woredas’, Col data will be ‘NaN’*

*#Define data as per each row/row-wise*

**data2 = [['Adis Ketema', 'Bole', 122541],**

**['Arada', 'Megenagna', 189625],**

**['Kolfe', 'Mexico', 154813],**

**['Kirkos', 'Piasa', 211467]]**

**df2 = pd.DataFrame(data=data2, index=['a', 'b', 'b', 'b'], columns=['Subcity', 'Downtown', 'Population'])** *#Unlike the above, here can’t supply column more length than single list size (col size)*

**df3 = pd.DataFrame({“num stud”: [192, 186, 154, 117, 89],**

**“num teacher”: [7, 8, 9, 10, 11],**

**“num course”: [1, 1, 2, 3, 4]},**

**index = pd.MultiIndex.from\_tuples([(‘Year 4’, ‘Sem I’), (‘Year 4’, ‘Sem II’), (‘Year 4’, ‘Sem III’),**

**(‘Year 5’, ‘Sem I’), (‘Year 5’, ‘Sem II’)],**

**names = [‘Year’, ‘Semester’]))** *#The tuples can be even created from python data structure such as zip(). Df3 is 2 dimensional (df3.ndim returns 2) but we can assume it is 3 dimensional*

**Reshaping Data**

**s.append(s2)** *#Extend the Series s*

**pd.melt(df)** *#Gather columns into rows with variable and value pairs (each colname will be the variable and each col value will be value*

**pd.melt(df, id\_vars=[‘colname’], value\_vars=[‘col2’, ‘col3’,…])** *#First col will be colname, second col ‘variable’ will has col data as col2, col3… vertically and third col ‘value’ will be col2 values and col3 values*

**pd.melt(df, id\_vars=[‘colname’], value\_vars=[‘col2’, ‘col3’,…], var\_name=’new\_colname\_for\_variable\_col’, value\_name=’new\_col\_name\_for\_value\_col’)** *#To give new col name for ‘variable’ and ‘value’ cols*

**df.pivot(columns=’col\_name\_as\_heading’, values=’col\_name\_as\_value’)** *#Spread rows into columns*

**df.stack()** *#Pivot a level of column labels (convert df to series by distributing all columns to index and create series*

**df.unstack()** *#Pivot a level of index labels (convert series to df by reversing stack. OR Make wide format by making rows as col heading for each colnames*

**df.unstack(‘index\_col’)** *#Select which col to make it wide*

**pd.concat([df1, df2,…])** *#Append rows of DataFrames (Vertically, default is axis=0). If ignore\_index=True is specified, index will be regenerated*

**pd.concat([df1, df2,…], axis=1)** *#Append columns of DataFrames (Horizontally)*

**df.sort\_values(‘col\_name’, ascending=False)** *#Default is ascending=True*

**df.sort\_index(ascending=False)** *#Default is ascending=True*

**df.set\_index(‘Subcity’)** *#Set the index*

**df.set\_index([‘col1’, ‘col2’,…])** *#Creates df by making listed column as multi-index and the rest column as data*

**df.reset\_index()** *#Reset index of DataFrame to row numbers, moving index to columns*

**df.drop(columns=[‘col1’, ‘col2’,…])** *#Drop columns from DataFrame*

**df.drop([0,1,5,…], axis=0)** *#Drop rows whose index is specified*

**pd.get\_dummies(df, columns=[‘obj\_col1’, ‘obj\_col2’,…], drop\_first=True)** *#To create one-hot encoding dummy variables (all data of listed in specified cols will be col heading and value will become boolean). ‘drop\_first=True’ used to avoid dummy variable trap or prevent the inclusion of redundant columns and avoid multicollinearity by omitting the first category in dummy (indicator) variable or one hot encoding creation.*

**df.[[‘col1’, ‘col2’, … ]].replace({‘old\_val1’: new\_val1, ‘old\_val2’: new\_val2,…}, inplace=True)** *#Used for manual label encoding. In this case, replacement will done across all listed cols****.***

**pd.cut(df[‘col’], bins=3)** *#Used to convert continuous variable into categorical variable. In this scenario, cut discretize into 3 equal-sized bins and create IntervalIndex numbers for those bins*

**pd.cut(df[‘col’], bins=3, labels=[‘label1’, ‘label2’, ‘label3’])** *#Rather assigning the IntervalIndex, use listed labels for each bins*

**bins = pd.IntervalIndex.from\_tuples([(0, 2), (3, 5), (6, 10)])**

**pd.cut(pd[‘col’], bins)** *#Use manually created bins as interval index. Values not covered by the IntervalIndex are set to NaN (in this scenario: 2.1, 5.9, 11,…).*

**pd.factorize(df[‘col’])** *#Encode the object as an enumerated type or categorical variable. Obtains a numeric representation for those distinct values of object. Returns tuple with array of converted code and array of unique objects to show code of object. Missing value is represented with -1*

**pd.to\_numeric(df[‘col’], errors=’coerce’)** *#Convert object to numeber, if non-numeric data found, assign NaN*

**pd.eval(“double\_age = df.age \* 2”, target=df)** *#Evaluate a Python expression as a string using various backends. In this scenario, scalar multiplication will be applied for the given column*

**pd.util.hash\_array(df[‘num\_col’])** *#Return hashed values for the given array of deterministic integers (needs np array)*

**pd.util.hash\_pandas\_object(df[‘col’])** *#Return hashed values for the given col*

**s.add\_prefix(‘label\_’)** OR **s.add\_suffix(‘\_label’)** *#For Series, the row labels are prefixed/suffixed*

**df.add\_prefix(‘label\_’)** OR **df.add\_suffix(‘\_label’)** *#For DataFrame, the column labels are prefixed/suffixed*

**Duplicate Data**

**s.unique()** *#Return unique values*

**df[‘w’].nunique()** *## of distinct values in a column*

**df.columns.is\_unique** *#Test if column values/labels are unique*

**df.duplicated()** *#Show boolean result if row is duplicated*

**df.duplicated([‘col1’, ‘col2’,…])** *#Show boolean result if those cols are duplicated*

**df.drop\_duplicates()** *#Remove duplicate rows (only considers columns)*

**df.drop\_duplicates([‘col1’, ‘col2’,…], keep=’last’)** *#Drop rows which is duplicated on the listed cols. keep=’last’ is used to remove all duplicates except the last entry or keep=’first’ used to remove all duplicates except the first entry*

**df.index.duplicated()** *#Returns numpy array which contains boolean data either index is duplicated or not*

**df.index** *#Returns Pandas RangeIndex to show row index range*

**df.index[0]** OR **df.index.get\_loc(0)** *#Access individual index*

**df.interpolate(method=’linear/time/index/pad/nearest/zero/slinear/quadratic/cubic/barycentric/polynomial/…’)** *#Fill NaN values using an interpolation method*

**Data Evaluation**

**df == 100** *#Each data is evaluated and returned boolean result*

**df.eq(100)** *#The same as the above*

**df.ne(val)** OR **df != val** *#Return True for not val for each data point, False otherwise*

**df == [val1, val2,…]** *#Evaluate arbitrary sequence with number of columns respectively*

**df[‘col’].le(val)** OR **df[‘col’] <= val**

**df[‘col’].lt(val)** OR **df[‘col’] < val**

**df[‘col’].ge(val)** OR **df[‘col’] <= val**

**df[‘col’].gt(val)** OR **df[‘col’] < val**

**df.all()** *#Returns each col’s evaluation to True if all data points are available( not 0 or empty) as per col or specific col, False if 0 or empty is found*

**df.any()** *#Returns False unless there is at least one element that makes the evaluation True*

**df.clip(lowerval, upperval)** *#Trim values at input threshold (value of data point will be replaced by lowerval if it is lower than lowerval or upperval if it is upper than upperval)*

**df.clip([col1lval, col1upval], [col2lval, col2upval],…)** *#Clip as per each col*

**Subset Observation of Rows**

**df[df.col\_name > 7]** OR **df[df[‘col\_name’] > 7]** OR **df.where(df[‘col\_name’] > 7)** *#Extract rows that meet the logical criteria (rows whose ‘col\_name’ is greater than 7 in this example)*

**df.sample(frac=0.5)** *#Randomly select fraction of rows of 50% from the total number of rows*

**df.sample(n=10)** *#Randomly select n rows*

**df[df[‘col’].isin([1, 2, 3, 4, 5])]** *#Select rows which satisfies the criteria*

**df[df[‘col’].str.contains(‘hello’)]** *#Select rows which satisfies the criteria*

**df.sample(n=10, replace=True, random\_state=42)** *#Replace is used for replacement (rows can be selected more than once regardless whether it is selected previously or not; random\_state uses to ensure reproducibility(not to reproduce randomly again and again) (if not stated, every time different row will be generated when the code is run).*

**df.nlargest(n, ‘col\_name’)** *#Select and order top n largest entries based on ‘col\_name’*

**df.nsmallest(n, ‘col\_name’)** *#Select and order bottom n smallest entries based on ‘col\_name’*

**df.head(n)** *#Select first n rows (5 if not stated)*

**df.tail(n)** *#Select last n rows (5 if not stated)*

**Subset Variables of Columns**

**df[[‘col1’, ‘col2’,…]]** *#Select multiple columns data with specific names*

**df[‘colname’]** OR **df.colname** *#Select single column data with specific name*

**df.filter(items=[‘col1’, ‘col2’,..])** *#Select listed columns only (use axis=1 to filter by row)*

**df.filter(like=’col\_chars’)** *#Filter columns with similar chars as col\_chars*

**df.filter(regex=’regex’)** *#Select columns whose name matches regular expression regex*

**df[‘col’].str.contains(‘regex’)**

**df[‘col’].str.startswith(‘regex’)**

**df[‘col’].str.endswith(‘regex’)**

**df[‘col’].str.replace(‘old’, ‘new)**

**df[‘col’].str.extract(‘(pattern)’)**

**regex (Regular Expressions) Examples**

**'\.'** *Matches strings containing a period '.'*

**'Length$'** *Matches strings ending with word 'Length'*

**'^Sepal'** *Matches strings beginning with the word 'Sepal'*

**'^x[1-5]$'** *Matches strings beginning with 'x' and ending with 1,2,3,4,5*

**'^(?!Species$).\*'** *Matches strings except the string 'Species*

**Using Query: query() allows Boolean expressions for filtering rows.**

**df.query(‘col1 > 500 and col2 == “value” and col3.str.startswith(“Eth”) or ~col4.str.endswith(“e”)’)** *#Filter data with col1 has >500 and col2 has “value” and col3 starts with “Eth” or col4 not ends with “e”*

**df.select\_dtypes([“dtype1”, “dtype2”,…])** OR **df.select\_dtypes(include=[“dtype1”, “dtype2”,…])** *#Select columns whose dtype is listed (float64, int16, int64, object, … etc)*

**Subsets: Rows and Cols**

*#df.loc[] and df.iloc[] to select only rows, only cols, or both*

**df.iloc[0:3]** *#Select rows whose index is between 0 and 2 (Starting is inclusive, ending is exclusive)*

**df.iloc[0, 2]** OR **df.iat[0, 2]** *#Select a value whose location is row index 0 and col index 2*

**df.iloc[:, [0, 2]]** *#Select all rows but only first col and third col (col can be listed as a list of index num)*

**df.iloc[a:b, x:y]** *#Select row from a to b and col x to y (b and y are exclusive)*

**df.loc[‘rowname’, ‘colname’]** OR **df.at[‘rowname’, ‘colname’]** *#Select single value by label (row name/number and col name)*

**df.loc[:, ‘col1’:’coln’]** *#Select all columns between col1 and coln (inclusive)*

**df.loc[:, [‘col1’, ‘col2’,…]]** *#Select only listed columns*

**df.loc[df[‘colname’] > 10, [‘col1’, ‘col2’]]** *#Select only col1 and col2 where colname is > 10*

**df.loc[:, (df>50).any()]** *#Select columns which has at least one of it’s value is greater than 50*

**df.loc[:, (df>50).all()]** *#Select columns which has every of it’s value is greater than 50*

**df.loc[:, df.isnull().any()]** OR **df.columns[df.isnull().any()]** *#Select columns which contain atleast 1 NaN*

**df.loc[:, df.isnotnull().all()]** *#Select columns without NaN*

**df.loc[‘a’:’b’, ‘x’:’y’]** *#Select row from a to b and col x to y (b and y are inclusive)*

**df.iat[row\_idx, col\_idx] = val** *#Change specific cell value by index*

**df.at[‘row\_lbl’, ‘col\_lbl’] = val** *#Change specific cell value by label*

**df[‘col\_lbl’].at[‘row\_lbl’] = val** OR **df[‘col\_lbl’].iat[row\_idx] = val** *#Chained is also possible*

**df.xs(key, axis)** *#Returns cross-section data (key=col index and axis= 0 for col 1 for row)*

**Summarize Data**

**df[‘w’].value\_counts()** *#Count each value of variable*

**len(df)** *## of rows in df*

**df.shape** *#Tuple of # of rows, # of cols*

**df.describe()** *#Basic descriptive & statistics for each col(or GroupBy)*

**df.describe(include=[‘O’])** *#Despite of the above, it describes the summary of columns whose dtype is not number (count, unique, top, freq)*

**Group by**

**dfg = df.groupby(by=[‘col1’, ‘col2’])** *#Create pandas DataFrameGroupBy for listed cols by grouping them for further data analysis and summary (for e.g: dfg.sum() shows each groups of col2 of col1 of summation)*

**dfg = df.groupby(level=0)** OR **df.groupby(level=[‘idx1’, ‘idx2’,…])** *#For multi-index grouping*

**dfg.head()** *#Show group data without grouping*

**dfg.first() OR dfg.last()** *#Shows first/last entries in all the group formed*

**dfg.get\_group((‘col1’s\_group\_val’,’col2’s\_group\_val’))** *#Finding original values contained based on the given tuples group (Tuple must be supplied if groupby is created based on multiple cols)*

**dfg.groups OR dfg.indices** *#List groups and their value location index as Pandas PrettyDict or dict respectively*

**dfg.size()** *#Compute group size for each group*

**dfg.ngroups** *#Returns number of groups as int*

**dfg.nth(x)** *#Extract xth row from each group*

**dfg.describe()** *#Summary information as per each group per each columns*

**dfg.agg([‘sum’, ‘mean’,…])** *#Use aggregate function to apply on each group*

**agg\_dic = {‘colname’: ‘agg\_func\_name’, ‘colname2’: [‘func\_name’, ‘agg\_func\_name2’,…],…}** *#For custom aggregate dictionary*

**Dates**

*#Pandas can manage dates and time as a point in time (timestamp) or as a span of time (period)*

**pd.Timestamp.min and pd.Timestamp.max** *#Returns range of date time value that pandas can work on*

**t = pd.Timestamp(‘2015-12-30’)** *#Create Pandas date*

**t = pd.Timestamp(‘2015-12-30 21:58:04.88’)** *#Create Pandas data and time (microsecond is optional)*

**p = pd.Period(‘2014-04-01’, freq=’M’)** *#Create period*

**ts = [‘2012-03-22’, ‘2022-04-09’,…]**

**s = pd.to\_datetime(pd.Series(ts))** *#Series of timestamps*

*#From non-standard strings to Timestamps*

**t = [‘09:08:55.7654-JAN092002’, ‘15:42:02.6589-FEB082016’]**

**s = pd.Series(pd.to\_datetime(t, format=”%H:%M:%S.%f-%b%d%Y”))** *#%B full month name, %m numeric month, %y year without century…*

*#Dates and timestamps and spans as indexes*

**date\_strs = ['2014-01-01', '2014-04-01', '2014-07-01', '2014-10-01']**

**dti = pd.DatetimeIndex(date\_strs)**

**pid = pd.PeriodIndex(date\_strs, freq='D')**

**pim = pd.PeriodIndex(date\_strs, freq='M')**

**piq = pd.PeriodIndex(date\_strs, freq='Q')**

**print(pid[1] – pid[0])** *#prints <90 days>*

**print(pim[1] – pim[0])** *#Prints <3 months>*

**print(piq[1] – piq[0])** *#Prints <1 quarter>*

*#PeriodIndex is preferable than DateTimeIndex unless less than second is necessary.*

**df.index = pd.period\_range(‘2015-01’, periods=len(df), freq=’M’)** *#Creates index for df as per every consecutive month starting 2015-01 with the length of periods (len(df)*

**df.index = pd.date\_range(‘2015-01’, periods=len(df), freq=’ME’)** *#Creates index for df as per every consecutive month starting 2015-01 with the length of periods (len(df) including the day information (last day of the month if day is not specified)*

**quarters = pd.period\_range(‘2002Q3’, periods=30, freq=’Q’)** *#Holds 30 consecutive quarters starting from the third quarter of 2002*

**dfp = df.to\_period(freq=’M’)** *#Used to convert the DateTimeIndex to PeriodIndex (Rather to show full date detail, it shows only specified period as an index / eliminating the day info from index)*

**dft = dfp.to\_timestamp()** *#Used to convert from PeriodIndex to DateTimeIndex. It defaults to the point in time at the start of the period (converting period to datetime in this scenario, the day information will be added but the first date of that month)*

**data = {**

**‘date’: pd.period\_range(start=’2015-01-01’, periods=7),**

**‘price’: [100, 102, 101, 104, 107, 108, 106, 110]**

**}**

**df = pd.DataFrame(data)** *#Create daily price list with a week days data*

**pi = pd.period\_range(‘1960-01-01’, ‘2015-12-31’, freq=’M’)** *#Generates period index as a list starting from 1960-01 to 2015-12 as per every month*

**df[‘Date’] = pd.to\_datetime(df[‘Date’])**

**february\_selector = (df.index.month == 2)** *#Creates boolean np array of size of df*

**february\_data = df[february\_selector]** *#Selects data from df by bolean operation (select only month 2 data in our case)*

**first\_quarter\_selector = df[(df.index.month >= 1) & (df.index.month <= 3)] ………**

**yearly\_data = df.groupby(df.index.year).sum()**

**pd.to\_datetime(df[‘datecol’]).dt.year**

**pd.to\_datetime(df[‘datecol’]).dt.month**

**pd.to\_datetime(df[‘datecol’]).dt.day**

**pd.to\_datetime(df[‘datecol’]).dt.day\_name()**

**pd.to\_datetime(df[‘datecol’]).dt.hour**

**pd.to\_datetime(df[‘datecol’]).dt.minute** *#...To extract the given information from the date col*

**pd.interval\_range(start=0, end=0)** *#Return a fixed frequency IntervalIndex [(0, 1], (1, 2], (2, 3], (3, 4], (4, 5]]. Intervals are all closed on the same side ((1, 2] means open boundary on the left side(1 is not included) and close boundary on the right side (2 is included))*

**pd.interval\_range(start=pd.Timestamp(‘2014-01-01’), end=pd.Timestamp(‘2014-01-04’))** *#Creates IntervalIndex as per each day (3 interval will be created)*

**pd.interval\_range(start=0, periods=4, freq=1.5)** *#Creates 4 IntervalIndex with the starting value of 0 and offset value of 1.5 [(0.0, 1.5], (1.5, 3], (3, 4.5], (4.5, 6]]. If start and end specified, freq is not necessary*

**pd.interval\_range(end=5, periods=4, closed=’both’)** *#Create 4 IntervalIndex backward with left and right boundary are closed [1, 2], [2, 3],…*

**pd.interval\_range(start=pd.Timestamp(‘2014-01-01’), periods=3, freq=’MS’)** *#Creates 3 IntervalIndex with DateOffset value of 1 month*

**df[‘col’].at\_time(‘12:00’)** *#Select values at particular time of any day. In this case, all data points with specified time will be selected*

**df[‘col’].between\_time(‘0:15’, ‘0:45’)** *#Select values whose time is between specified range of time*

**Summary functions:**

*#Produce single values for each groups(features) with a data type of pandas Series if it has single column data with index(features or each groups)*

*#Use axis=0 for col-wise operations (which is default) and axis=1 for element-wise operations(row-wise)*

**df.describe(), df.sum(), df.cumsum(), df.cummax(), df.cummin(), df.cumprod()** *#Cummulative result***, df.count(), df.median(), df.min(), df.max(), df.mean(), df.abs(), df.add(o), df.div(o), df.mul(o),** *#Scalar or vector operation on each value (default is row-wise; use axis=0 to make it col-wise***, df.dot((l, m, n, o,….))** *#dot product***, df.diff(), df.var(), df.std(), df.cov(numeric\_only=True), df.corr(numeric\_only=True), df.idxmin(), df.idxmax()** *#Return min or max values index number***, df.kurt()** *#Kurtosis over cols (def)***, df.sem()** *#Standard error of mean***, df.mad()** *#Mean absolute deviation*

**df.quantile()** OR **df.quantile([0.25, 0.5, 0.75, 0.8,…])** *#Returns df if quantile list is supplied*

**df.drop(columns=’label’).quantile()** OR **df.drop(columns=[‘col1’, ‘col2’, ‘col3’,...])** *#To exclude column or list of columns from group function*

**sums = {c: df[c].sum() for c in df}** *#Create dictionary which contains col\_name as key, sum of cols value as a value (for c in df returns str of each col\_name).*

**pd.crosstab(index=df[‘col’], columns=df[‘row\_to\_col’])** *#Cross-tabulation (frequency count)*

**Applying Functions**

**func = lambda x: x\*2**

**df.apply(func)** *#Apply function to each object/group/feature*

**df[‘colname].apply(func)** *#Apply func to each value of column*

**df.applymap(func)** *#Apply function element-wise*

**Handling Missing Data**

**df.dropna()** *#Drop rows with any column having NA/null data*

**df.dropna(axis=1)** *#Drop columns containing missing values*

**df.dropna(how=’all’)** *#Drop all NaN row*

**df.dropna(thresh=2)** *#Drop 2+ NaN in row*

**df.dropna(df[‘col’].notnull())** *#Only drop row if NaN in a specified col*

**df.fillna(value)** *#Replace all NA/null data with value*

**df[‘col’].fillna(value)** *#Only selected col*

**df.fillna(df.mean())** *#Fill NaN values with a predetermined value*

**df.replace(‘oldval’, ‘newval’)** *#Replace values with others*

**Make New Columns**

**df[‘newcol’] = df.col1 \* df.col2** *#Add single column*

**df.assign(newcol1=lambda df: df.col1 \* df.col2, newcol2=val, newcol3=”val”,...)** *#Compute and append one or more new columns*

**df.rename(columns={‘old\_col\_name1’: ‘new\_col\_name1’, ‘old\_col\_name2’: ‘new\_col\_name2’,…})** OR

**df.columns = [‘new\_col1\_name’, ‘new\_col2\_name’,…]** *#Rename column’s labels*

**df = df[[‘col3’, ‘col2’, ‘col1’,…]]** *#Reorder col location([[]] represents pd dataframe where as [] represents pd Series)*

**df[df.columns[[0, 2, 4…]]]** *#Select by col index*

**df[df.columns[:-1]]** *#All but last col*

**df[‘new\_col’ = df[‘existed\_col’].where(df[‘existed\_col’] > 0, other=0)** *#Create new column based on criteria; assign a value described at other if criteria not met(where and other can be pd Series)*

**df[‘col’].to\_datetime()** *#Convert the column value to date format*

**df[‘new\_col’] = df.sum(axis=1)** *#Row summation to to new col*

**Combine Dataset**

*#For different df but have common id (‘c1’).* **df1 = pd.DataFrame({‘c1’: [‘A’, ‘B’, ‘C’], ‘c2’: [1, 2, 3]})**

**df2 = pd.DataFrame({‘c1’: [‘A’, ‘B’, ‘D’], ‘c3’: [T, F, T]})**

**pd.merge(df1, df2, how=’left’, on=’c1’)** *#Join matching rows from df2 to df1. (‘A’, ‘B’, ‘C’) OR*

**pd.merge(df1, df2, how=’left’ left\_on=’col1’, right\_on=’col2’)** *#If pk of both df are founds in different col names*

**pd.merge(df1, df2, how=’right’, on=’c1’)** *#Join matching rows from df1 to df2. (‘A’, ‘B’, ‘D’)*

**pd.merge(df1, df2, how=’inner’, on=’c1’)** *#Join data. Retain only rows in both sets (‘A’, ‘B’)*

**pd.merge(df1, df2, how=’outer’, on=’c1’)** *#Join data. Retain all values, all rows (‘A’, ‘B’, ‘C’, ‘D’)*

**df1[df1.c1.isin(df2.c1)]** *#All rows in df1 that have a match in df2 (‘A’, ‘B’)*

**df1[~df1.c1.isin(df2.c1)]** *#All rows in df1 that do not have a match in df2 (‘C’)*

*#For similar df with the same cols,*

**df1 = pd.DataFrame({‘c1’: [‘A’, ‘B’, ‘C’], ‘c2’: [1, 2, 3]})**

**df2 = pd.DataFrame({‘c1’: [‘B’, ‘C’, ‘D’], ‘c2’: [2, 3, 4]})**

**pd.merge(df1, df2)** *#Rows that appear in both df1 and df2 (intersection) (‘B’, ‘C’)*

**pd.merge(df1, df2, how=’outer’)** *#Rows that appear in either or both df1 and df2 (union) (‘A’, ‘B’, ‘C’, ‘D’)*

**pd.merge(df1, df2, how=’outer’, indicator=True)**

**.query(‘\_merge == “left\_only”’)**

**.drop(columns=[‘\_merge’])** *#Rows that appear in df1 but not df2 (setdiff). indicator adds column ‘\_merge’ to show which row belongs to which df(‘left\_only’, ‘right\_only’, ‘both’) (‘A)*

**Plotting**

**df.plot.hist()** *#Histogram for each column*

**df.plot.scatter(x=’w’, y=’h’)** *#Scatter chart using pairs of points*

**df.boxplot(column=’colname’, grid=False)** *#Used for outlier detection*

**df.plot.bar(x=[‘colx1’, ‘colx2’,…], y=[‘coly1’, ‘coly2’,…])** *#Plot bar*

**grouped.sum().plot.pie(y=’ycols’, subplots=True)** *#Create pie chart*

**df.plot.area(x=’x-axis’, y=’ycols’, stacked=True)**

**df.plot.density()**

**df.plot.hexbin(x=’xval’, y=’yval’)**

**df.plot.line(x=’xval’, y=’yval’)**

**df.plot.pie()**

**Iterators**

**df.items()** *#(column-index, col to Series) pairs generator. (Series contains row index with value)*

**list(df.items())[0]** *#List specific entry (col-index, Series) pairs based on column-index (0 is the first column index)*

**list(df.items())[0][0]** *#Display the col-index part of the first pair*

**list(df.items())[0][1]** *#Display the Series part of the first pair*

**list(df.items())[0][1][0]** *#Display elements of the first Series of first pair (first element in this case)*

**df.iterrows()** *#(Row-index, row to Series) pairs*

*#Operations are the same with the df.items()*

**list(df.iterrow()[0][1].colname** OR **list(df.iterrows()[0][1][‘col name’]** *also possible*

**df.itertuples()** *#Creates maps*

*#Operation is the same above but with 2 dimension only*

**Read and Write to df with Different Format**

**df = pd.read\_csv(‘path/file\_name.csv’)**

**df = pd.read\_csv(‘path/file\_name.csv’, header=None, nrows=5)**

**df = pd.read\_csv(‘path/file\_name.csv’, header=0, index\_col=0, quotechar=’”’, sep=’:’, na\_values = [‘na’, ‘-’, ‘.’, ‘’], usecols=[‘col1’, ‘col2’])[[‘col2’, ‘col1’]]** *#header: used to represent which row is assumed as a header index/label. index\_col: used to represent which col is assumed as a row index/label. quotechar: used to identify which character is used for quoting in csv. sep: used to identify what separator is csv file used. na\_values: used to identify what value is used to represent NaN in csv file. Usecols: is used to tell which col will be read. Can be determined as col label or col index. (to reorder the cols, specify order of cols as slicing*

*#Other options:*

**dtype={‘col1’: np.float64, ‘col2’: ‘Int64’, ‘col3’: ‘str/object’}** *#Specify dtype for each col*

**engine = {‘c, ‘python’, ‘pyarrow’}** *#Select parser engine to use. C and pyarrow are faster, while python is more feature-complete. Multithreading is supported by pyarrow*

**converters={‘col1’: func1, ‘col2’: func2,…}** *#Functions for converting values in specified cols*

**df = pd.to\_csv(‘path/file\_name.csv’, encoding=’utf-8’, index=False)** *#index used to avoid writing row indices*

**workbook = pd.ExcelFile(‘path/file\_name.xlsx’)**

**df = pd.read\_excel(workbook, ‘Sheet1’)** *#Read the Sheet1 of the excel file ‘file\_name.xlsx’*

**df.to\_excel(‘file\_name.xlsx’, sheet\_name=’Sheet1’)** *#Create and save the dataset to file\_name.xlsx as Sheet1*

*#Or if the sheet name is unknown or wants to load all sheets:*

**d = {}**

**for sheet\_name in workbook.sheet\_names:**

**df = workbook.parse(sheet\_name)**

**d[sheet\_name] = df**

**df = pd.read\_json(‘file\_name.json’)** *#Read json file*

**df = pd.read\_sql(‘file\_name.sql’)** *#Read sql file*

**df.to\_sql(‘file\_name.sql’)**

**df = pd.read\_html(‘file\_name.html’)** *#Parse html to find all tables*

**Import pymysql**

**from sqlalchemy import create\_engine**

**engine = create\_engine(‘mysql+pymysql://’ + ‘USER:PASSWORD@HOST/DATABASE’)**

**df = pd.read\_sql\_table(‘table\_name’, engine)**

**df = pd.DataFrame(np.random.rand(50, 5))** *#Generate fake data with 50 rows by 5 cols.*

**df.to\_dict()** *#Convert df to dictionary*

**df.to\_string()** *#Convert df to string*

**pd.to\_pickle(df, ‘file\_name.pkl’)** *#To serialize the dataframe (converting object to byte stream)*

**pd.read\_pickle(‘file\_name.pkl’)** *#To deserialize the dataframe (converting byte stream to object)*

**pd.read\_clipboard()** *#Read data from clipboard and convert into dataframe (Copy table from any source and just run this code, the copied data will be populated as a dataframe)*

**pd.DataFrame.to\_clipboard()** *#Copy object to the system clipboard, so that it can pasted into Excel or other place*

**pd.read\_xml(‘file\_name.xml’)** *#Read the xml source*

**pd.DataFrame.to\_xml()** *#Convert dataframe to xml*

**Info and Conversion**

**pd.\_\_version\_\_** *#Display version information of Pandas*

**df.info()** *#Check information about the data*

**df.values** *#Convert df to NumPy array*

**df.dtypes** *#Display data types of all columns*

**df[‘colname’].astype(‘int32’)** OR **df[‘colname’].astype(int)** *#Change the data type of column colname to specified dtype*

**df.axes** *#List row and col indices (Returns list as a result)*

**df.empty** *#True for empty DataFrame*

**df.ndim** *#Number of axes /dimensions*

**df.size** *#Total number of data (row\_count \* col\_count)*

**df.copy()** *#Copy a DataFrame*

**df.rank()** *#Rank each col*

**df.columns** *#Returns Pandas Index contains list of name of cols*

**df.columns[0]** *#Access each colname as str/object*

**df.columns.tolist()** *#Convert Pandas Index of col names to list*

**df.columns.values** *#Convert Pandas Index of col names to numpy array*

**df.round(decimals=0)** *#Rounding the df to the specified decimal places*

**df[‘newcol’] = df[‘existed\_col’].rolling(window=3).mean()** *#Rolling window (moving window) applies a calculation over a fixed-size subset of the data which rolls forward. Window sliding for only for the last n data points (3). (Used for short-term time series analysis like detecting trends over a recent period)*

**df[‘newcol’] = df[‘existed\_col’].expanding().mean()** *#Expanding window starts from the first data point and includes more data as moved forward. Calculation is applied to all data points starting from the first to the current data point. (Suitable for cumulative time series analysis like tracking the overall trend or long-term average)*

**s.memory\_usage()** *#Bytes of memory consumed*

**df.memory\_usage()**

**help(pd.Series.loc)** *#Displays documentation on the given object*

**wget, curl, and aria2c Cheat Sheet. Compiled by Sisay Zeleke tosisay@gmail.com**

**Wget Download Options**

**wget http://domain\_name** *#Download the index page of given website*

**wget -c|–-continue url/file.iso***#Resume a partial download*

**wget url/A{1..12}.jpg***#Download A1.jpg, A2.jpg, up to A12.jpg*

**wget -m|–-mirror http://domain\_name***#Download the entire url site*

**wget -d|–-debug http://domain\_name***#View HTML headers*

**wget url -t|--tries=num***#Try num times if failed to download*

**wget url -t|–-tries inf***#Try all but 404 downloads infinite times until successfully downloaded. Default: 20*

**wget url -nc|--no-clobber***#If file is already existed, skip downloading it*

**wget url –-spider** *#Do not download files, just crawl the site. Used to check for the existence of file/domain*

**wget url -T|–-timeout 30** *#Stop an unsuccessful download attempt within 30 seconds. Default: 900 seconds. It combines -–dns-timeout for DNS lookup time, --connect-timeout for TCP connection time, and –-read-timeout for idle time*

**wget url –-limit-rate 50k** *#Limit download rate to 50k per second*

**wget url -w|–-wait 5** *#Wait 5 seconds between downloads*

**wget url -w|–-wait 5 -–random-wait** *#Multiply –wait value by a number from 0.5 to 1.5*

**wget url –-user foo –-password bar** *#Send user name foo and password bar*

**wget url –-ask-password** *#Prompt for interactive password*

**Wget url -N|–-timestamping** *#Do not re-retrieve files unless newer than local*

**wget url –-background** *#Send to background immediately after startup*

**wget url --compression=TYPE** *#Choose compression from auto, gzip, or none*

**Wget Logging and Directory/File Options**

**wget url -O|-–output-document -** *#Display output in terminal*

**wget url -o|–-output-file=fname** *#Log all messages to logfile fname*

**wget url -q|–-quiet** *#Turn off output information*

**wget url -nv|–-no-verbose** *#Turn off verbose without being completely quiet (display completion status only, not the detail)*

**wget url --show-progress** *#Display the progress bar in any verbosity mode*

**wget i|–-input-file=file\_location** *#Download urls found in specified local or external file*

**wget url –-rejected-log=logfname** *#Logs all url rejections to logfile*

**wget url -nd|–-no-directories** *#Don’t create directories*

**wget url -x|–-force-directories** *#Create a directory with domain name and store files inside it*

**wget url -P|--directory-prefix=dir\_name** *#Save files to dir\_name*

**wget url -S|--server-response** *#Print server response*

**wget http://domain\_name -O|-–output-document foo.html** *#Save index.html as foo.html*

**Wget HTTP Options**

**wget http://domain\_name “-U|–-user-agent=User:Agent: mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124”** *#Identify as AGENT instead of Wget/VERSION (appear as the Chrome browser running on Windows 10)*

**wget url –-no-cache** *#Disallow server-cached data*

**wget url –-proxy-user=USER --proxy-password=PASS** *#Set USER and PASS as proxy username and password*

**wget url –-referer=URL** *#Include ‘Referer:URL’ header in HTTP request*

**wget url –-save-headers** *#Save the HTTP headers to file*

**wget url –-no-cookies** *#Don’t use cookies*

**Wget HTTPS (SSL/TLS)Options**

**wget url –-secure-protocol=PR** *#Choose secure protocol, one of auto, SSLv2, SSLv3, TLSv1, TLSv1\_1, TLSv1\_2 and PFS*

**wget url –-no-check-certificate** *#Don’t validate the server’s certificate and ignore certificate errors*

**wget url –-certificate=FILE** *#Client certificate file*

**wget url –-private-key=FILE** *#Private key file*

**Wget FTP Options**

**wget url –-ftp-user=USER --ftp-password=PASS** *#Set ftp user and password*

**wget url –-no-glob** *#Turn off FTP file name globbing*

**Wget Recursive Options**

**wget url -r|–-recursive** *#Specify recursive download*

**wget url -l|–-level=NUM** *#Maximum recursion depth*

**wget url -A|–-accept=LIST** *#Comma-separated list of accepted extensions*

**wget url -R|–-reject=LIST** *#Comma-separated list of rejected extensions*

**wget url -D|–-domains=LIST** *#Comma-separated list of accepted domains*

**wget url –-exclude-domains=LIST** *#Comma-separated list of rejected domains*

**wget url -I|–-include-directories=LIST** *#List of allowed directories*

**wget url -X|–-exclude-directories=LIST** *#List of excluded directories*

**wget url -np|–-no-parent** *#Don’t ascend to the parent directory*

**+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++**

**Curl Web Browsing Options**

**curl url** *#Return the source file of the URL. Assumed GET request if not specified*

**curl url1 url2** *#Return multiple source files from multiple URL*

**curl url -e|--referer url** *#Some website requests referrer link in order to be accessed by client*

**curl ftp://domain/dir** *#Get an FTP directory listing*

**curl url/dir -l|--list-only** *#List contents of the directory*

**curl url -L|--location** *#Redirect query as specified by HTTP status response code 3xx. This means if URL is not found and the page is moved to another location and the new location is described in HTTP status response, it redirects to the new address and return it*

**curl url -I|--head** *#Fetch HTTP headers of the URL*

**curl url -i|--include** *#Complete header information as well*

**curl url -v|--verbose** *#Make the operation more talkative*

**curl url -s|--silent** *#Do not show progress meter or errors*

**curl url -S|--show-error** *#Show errors*

**curl url -E|--cert FILE** *#Use file for certificate*

**curl url --cert-type TYPE** *#Use der|pem|eng|p12 as certificate type*

**curl “https://bit.ly/2ye45g3” -I|--head -L|--location** *#Expand a shortened URL*

**curl url \_I|--head -S|--show-error** *#Check whether the site is down*

**curl url -x|--proxy proxyaddr:portno** *#Access the URL through specified proxy info*

**curl url -u|--user uname:pwd** *#Use the specified user name and pass to access the URL. If pwd is not specified, curl will prompt for it*

***curl url -m|--max-time 10*** *#Limit the amount of time a request may take*

***curl url --connect-timeout 3*** *#Set maximum second that the connection phase may take*

***curl url --trace-time*** *#Display all information*

**Curl Downloading Options**

**curl url -o|-–output NAME** *#Outputs the URL to NAME file*

**curl url -A|–-user-agent “Mozilla/5.0 (X11; Linux x86\_64; rv:60.0) Gecko/20100101 Firefox/81.0”** *#Specify user agent (Mozilla version 81 in this case)*

**curl url -O|--remote-name** *#Download a file from URL, saving the file without changing its name*

**curl url --remote-name-all** *#Use the remote file name for all URLs*

**curl url -O|--remote-name -o|--output foo.html** *#Download a file from URL and save with the given name*

**curl url -O|--remote-name -C|–-continue-at -** *#Continue a partial download of a file stated in URL*

**curl url/pic[0-9].jpg** *#download all sequentially-numbered files*

**curl “url{file1,file2,file3,...}” -o|--output “File\_#1.ext”** *#Download files from multiple locations and name them according to the format File\_file1.ext, File\_file2.ext,…. URL structure should like url/file1, url/file2, url/file3,… etc*

**curl “www.{domain1,domain2,domain3,…}.org” -o|--output “File\_#1.ext”** *#Download files from multiple domains and name them according to the format File\_domain1.ext, File\_domain2.ext,… etc*

**curl “url/[150-155]/[11-13]-book.{txt,zip}” -o|--output “bk#1\_#2.#3”** *#Download a sequence of files and outputs bk150\_11-book.txt, bk150\_11-book.zip, bk151\_11-book.txt, bk151\_11-book.zip,… bk155\_13-book.txt, bk155\_13-book.zip*

**curl url -C|–-continue-at <offset>** *#Resumed transfer offset*

**curl url –-compressed** *#Request compressed response*

**curl url –-max-filesize <bytes>** *#Maximum file size to downloaded*

**curl url -m|–-max-time <seconds>** *#Maximum time allowed for transfer*

**curl url –-no-clobber** *#Do not overwrite files that already exist*

**curl url -#|–-progress-bar** *#Display transfer progress bar*

**curl url –-no-progress-meter** *#Do not show the progress meter*

**curl -K|–-config urls.txt -Z|–-parallel** *#Perform transfers in parallel urls.txt contains multiple URLs to be downloaded*

**curl -K|–-config urls.txt –-parallel-immediate** *#Prioritize creating new connections instead of waiting for other connections to finish*

**curl -K|–-config urls.txt –-parallel-max <num>** *#Maximum concurrency for parrallel transfers*

**curl url –-rate <max request rate>** *#Request rate for serial transfer*

**curl url –-raw** *#Do HTTP “raw”, no transfer encoding*

**curl url --remove-on-error** *#Remove output file on errors*

**curl url --retry <num>** *#Retry request if transient problems occur*

**curl url --retry <num> --retry-all-errors** *#Retry all errors*

**curl url --retry <num> --retry-connrefused** *#Retry on connection refused*

**curl url --retry <num> --retry-delay <seconds>** *#Wait time between retries*

**curl url --retry <num> --retry-max-time <seconds>** *#Retry only within this period*

**curl url -s|--silent** *#Silent mode*

**Curl GET Options**

**curl url -X|–-request GET** *#Fetch the HTML source of the URL and output it in the terminal console*

**curl www.example.com?name=alice&age=25** *#Post data via GET request*

**curl ‘url/viewdatax/endpoint/view?secret=skjc2sdf3n3df2oewrkl3sfs5sdnd’ -X|--request GET -H|–-header ‘Content-Type: application/json’** *#Get all data from the viewdatax app with the given secret string and content type header as query parameters. The expected result is a JSON object containing all documents*

**curl ‘url/viewdata/endpoint/view?secret=skjc2sdf3n3df2oewrkl3sfs5sdnd&id=3hrm9db7k4’ -X|--request GET -H|–-header ‘Content-Type: application/json’** *#Get only the data whose id is given by id= from the viewdata app with the given secret string and content type header as query parameters. The expected result is single JSON object of a given id*

**curl -K|--config myconfig.conf** *#Use a configuration file for repeated tasks. myconfig.conf file may contain: url=URL -–compressed -–head –-no-progress-meter user-agent =”AGENT” \n url=URL2 options for url2… etc*

**Curl POST Options**

**curl ‘url/app/datax/endpoint/data/v1/action/insertOne’ -X|–-request POST -H|--header ‘Content-Type: application/json’ -H|–-header ‘api-key: A3gDdGyT68kmUgJsxgVty7Vjf575fEREe346IhKr57’ -–data-raw ‘{“document”: {“name”: “Alice Bob”, “age”: 25, “greeting”: “Greetings, everyone!”}}’** *#Upload data via API.*

**curl ‘url/app/datax/endpoint/data/v1/action/findOne’ -X|–-request POST -H|--header ‘Content-Type: application/json’ -H|–-header ‘api-key: A3gDdGyT68kmUgJsxgVty79ERygVjf575fEREe346IhKr57’ -–data-raw ‘{“filter”: {“name”: “Alice Bob”}}’** *#Inquire data via API for the document with the given key-value pair filter.*

**curl ‘url/app/datax/endpoint/data/v1/action/deleteOne’ -X|–-request POST -H|--header ‘Content-Type: application/json’ -H|–-header ‘api-key: A3gDdGyT68kmUgJsxgVty79ERygVj3gREe346IhKr57’ -–data-raw ‘{“filter”: {“\_id”: {“$oid”: “vi3m66vgejg06u3hf7”}}}’** *#Delete data via API for the document with the given id filter.*

**curl url -F|--form upload=@filename -F|--form press=OK** *#To upload file via form post content-type:multipart/form-data*

**curl url -T|--upload-file FILENAME** *#Upload file using PUT request*

**Curl API Interaction**

**curl “url/api/v1/projects”** *#Query an API endpoint*

**curl “url/api/v1/endpoint” -H|--header “Auth-Token:$DB\_APP\_TOKEN”** *#Pass a header to a server URL. A header is a field of an HTTP request or response that passes additional context and metadata about the request/response. In this example, the header is an authorization token.*

**curl “https://docs.google.com/forms/d/e/[GoogleFormID]/formResponse” -d|--data “ABC 123”** *#Send URL-encoded raw data to an API endpoint, in this case a Google Form*

**curl “https://docs.google.com/forms/d/e/[GoogleFormID]/formResponse” -d|--data “ABC 123” > output.html** *#Send URL-encoded raw data to an API endpoint, in this case a Google Form and output to output.html data returned from the server*

**curl “https://docs.google.com/forms/d/e/[GoogleFormID]/formResponse” > output.html -F|--form “emailAddress=mymail@domain.com” -F|–-form “submit=Submit”** *#Emulate sending email addr and pressing the Submit button. The output file will have a filled email address field*

**curl “https://docs.google.com/forms/d/e/[GoogleFormID]/formResponse” > output.html -F|--form “entry.123456=<Documents/myfile/lists.txt”** *#Send to an API endpoint(Google Form) the file contents of lists.txt to the parameter entry.123456. < means sending data to server and > means receiving data from server. Output file will show the file contents in the corresponding field.*

**curl “https://docs.google.com/forms/d/e/[GoogleFormID]/formResponse” -F|--form “entry.123456=<Documents/myfile/lists.txt” -F|--form “emailAddress=mymail@domain.com”** *#Send more than one piece of data to the given API endpoint(Google Form). This command sends over the email and lists file specified. Output for this command will be in the terminal.*

**curl “https://docs.google.com/forms/d/e/[GoogleFormID]/formResponse” -d|--data “entry.123456=<Documents/myfile/lists.txt&emailAddress=mymail@domain.com”** *#Similar with the above but with different approach.*

**curl “image-resizer.com/resize-image” > output.html -F|--form “input=@mypic.jpg”** *#Send a jpg file as form data to the given API endpoint. @ represents relative/current working dir. The output file will show the returned options by the API*

**+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++**

**Aria2c Downloading Options**

**aria2c uri** *#Download the specified URI*

**aria2c “www.server1.com/file1.zip” “www.server2.com/file1.zip”** *#Download the file from multiple servers*

**aria2c uri -d|-–dir=<DIR>** *#Download the specified URI and store the file in the given directory*

**aria2c uri -d|-–dir=<DIR> -o|--out=<FILENAME>** *#Download and rename with the given FILENAME inside DIR*

**aria2c -i|–-input-file=<FILE>** *#Download the URIs listed in file. Multiple sources for a single entity can be specified using TAB chars. Options can be specified after each URL*

**aria2c uri -l|-–log=<LOG>** *#File name for the log file*

**aria2c uri -j|-–max-concurrent-downloads=<num>** *#Set the maximum number of parallel downloads for every queue item*

**aria2c uri -V|--check-integrity=[true|false]** *#Check file integrity by validating the piece hashes or a hash of entire file (default is false). This option has effect only BitTorrent, Metalink downloads with checksums or HTTP(S)/FTP downloads with –-checksum option. If piece hashes are provided, this option can detect damaged portions of a file and re-download them. If a hash of entire file is provided, hash check is only done when file has been already downloaded. This is determined by file length. If hash check fails, file is re-downloaded from scratch. If both piece hashes and a hash of entire file are provided, only piece hashes are used.*

**aria2c uri -c|--continue=[true|false]** *#Continue downloading a partially downloaded file. Use this option to resume a download started by another program which downloads files sequentially from the beginning. This option is applicable to HTTP(S)/FTP downloads.*

**Aria2c HTTP/FTP/SFTP Options**

**aria2c uri –-all-proxy=<http://USER:PASS@HOST:PORT>** *#Use a proxy server for all protocols. To override a previously defined proxy, use “”*

**aria2c uri –-all-proxy=<http://HOST:PORT> –-all-proxy-user=<USER> –-all-proxy-passwd=<PASS>** *#Set USER and PASS for --all-proxy option*

**aria2c uri –-connect-timeout=<SEC>** *#Set the connection timeout in seconds (default: 60) to establish connection to HTTP/FTP/proxy server. After the connection is established, this option has no effect.*

**aria2c uri –-dry-run=[true|false]** *#Just checks whether the remote file is available and do not download data (default: false)*

**aria2c uri –-lowest-speed-limit=<SPEED>** *#Close connection if download speed is lower than or equal to this value (bytes per sec). 0 means no limitation. Use K for KiB, M for MiB. This option doesn’t affect BitTorrent downloads (default: 0)*

**aria2c uri -m|–-max-tries=<NUM>** *#Set number of tries. 0 means unlimited. (default: 5)*

**aria2c uri -m|–-max-tries=<NUM> --max-file-not-found=<NUM>** *#If file not found happen, force the download to fail because it sets to 0 (means unlimited and it is default). It is counted toward --max-tries*

**aria2c uri -x|–-max-connection-per-server=<NUM>** *#Maximum number of connections to one server for each download (default: 1)*

**aria2c uri -k|–-min-split-size=<SIZE>** *#Determine size of each chunk. Each source determined by --split is serves different chunks. Possible value range: 1M-1024M (default: 20M)*

**aria2c uri -s|-–split=<NUM>** *#Download a file using NUM connections. The number of connections to the same host is restricted by --max-connection-per-server option (default: 5). Some Metalinks regulate the number of servers to connect. If Metalink defines the maxconnections attribute lower than NUM, maxconnections value will be utilized*

**aria2c uri -–retry-wait=<SEC>** *#Waits between retry when HTTP server returns a 503 response (default: 0)*

**aria2c uri –-stream-piece-selector=[default|inorder|random|geom]** *#Specify piece selection algorithm used in HTTP/FTP download. A piece is a fixed length segment which is downloaded in parallel in a segmented download. (default: default). default: Select a piece to reduce the number of connections established since establishing a connection is an expensive operation, it is reasonable default behavior. Inorder: Select a piece closest to the beginning of the file. This is useful for viewing movies while downloading. --enable-http-pipelining option may be useful to reduce re-connection overhead. --min-split-size option should be reasonable because it affects the option. random: Select a piece randomly. --min-split-size can affect the option. geom: When starting to download a file, select a piece closest to the beginning of the file like inorder, but then exponentially increases space between pieces. This reduces the number of connections established, while at the same time downloads the beginning part of the file first. This is useful for viewing movies while downloading.*

**aria2c uri -t|–-timeout=<SEC>** *#Set timeout in seconds (default: 60)*

**Aria2c HTTP Specific Options**

**aria2c uri –-ca-certificate=<FILE>** *#Use the certificate authorities in FILE to verify the peers. The certificate file must be in PEM format and can contain multiple CA certificates. Use --check-certificate option to enable verification.*

**aria2c uri –-certificate=<FILE>** *#Use the client certificate in FILE. The certificate file must be either in PKCS12 or PEM format. When using PEM, --private-key should be specified.*

**aria2c uri –-check-certificate=[true|false]** *#Verify the peer using certificates specified in --ca-certificate option (default: true)*

**aria2c uri –-http-no-cache=[true|false]** *#Send Cache-control: no-cache and Pragma: no-cache header to avoid cached content. (default: false)*

**aria2c uri –-http-user=<USER> --http-passwd=<PASSWD>** *#Set HTTP USER and PASSWD*

**aria2c uri –-http-proxy=<http://USER:PASSWD@HOST:PORT>** *#Use a proxy server for HTTP. To override previously defined proxy, use “”*

**aria2c uri –-http-proxy-user=<USER> --http-proxy-passwd=<PASSWD>** *#Set HTTP proxy USER and PASSWD*

**aria2c uri –https-proxy=<PROXY> –-https-proxy-user=<USER> --https-proxy-passwd=<PASSWD>** *#Set HTTPS proxy information*

**aria2c uri –-private-key=<FILE>** *#Use the private key in FILE. The private key must be decrypted and in PEM format.*

**aria2c uri –-referer=<REFERER>** *#Set an HTTP referrer. If \* is given, the download URI is also used as the referrer.*

**aria2c uri –-enable-http-pipelining=[true|false]** *#Enable HTTP/1.1 pipelining. (default: false)*

**aria2c uri –-header=<HEADERINFO1> --header=<HEADERINFO2>…** *#Append HEADERINFO to HTTP request header. Multiple header info can be applied*

**aria2c uri –-load-cookies=<FILE>** *#Load Cookies from FILE using the Firefox, Chrome, or Netscape format*

**aria2c uri –-save-cookies=<FILE>** *#Save Cookies to FILE in Mozilla, Netscape format*

**aria2c uri -U|–-user-agent=<USER\_AGENT>** *#Set user agent for HTTP(S) downloads.*

**Aria2c FTP/SFTP Specific Options**

**aria2c uri –-ftp-user=<USER> --ftp-passwd=<PASSWD>** *#Set FTP USER and PASSWD*

**aria2c uri –-ftp-pasv=[true|false]** *#Use passive mode in FTP if true is given. Else active mode will be activated which is default.*

**aria2c uri –ftp-proxy=<PROXY> –-ftp-proxy-user=<USER> --ftp-proxy-passwd=<PASSWD>** *#Set FTP proxy information*

**Aria2c BitTorrent Specific Options**

**aria2c -T--torrent-file=\*.torrent** *#The path to the .torrent file. This option is not required to specify torrent file.*

**aria2c \*.torrent –-bt-detach-seed-only=[true|false]** *#With concurrent downloads, exclude an active download which enters seed mode(completed downloads) (default: false)*

**aria2c \*.torrent –-bt-enable-lpd=[true|false]** *#Enable Local Peer Discovery. If a private flag is set in a torrent, this option won’t work (default: false)*

**aria2c \*.torrent –-bt-exclude-tracker=<URI>[,… ]** *#Comma separated list of BitTorrent tracker’s announce URI to remove. Matching chars such as \* can be used*

**aria2c \*.torrent –-bt-force-encryption=[true|false]** *#Requires BitTorrent message payload encryption with arc4.It deny legacy BitTorrent handshake and only use Obfuscation handshake and always encrypt message payload. (default: false)*

**aria2c \*.torrent –-bt-max-open-files=<NUM>** *#Specify maximum number of files to open in multi-file BitTorrent/Metalink download globally. (default: 100)*

**aria2c \*.torrent –-bt-max-peers=<NUM>** *#Specify the maximum number of peers per torrent. 0 means unlimited. (default: 55)*

**aria2c \*.torrent –-bt-metadata-only=[true|false]** *#Download metadata only. The file(s) described in metadata will not be downloaded. This option has effect only when BitTorrent Magnet URI is used. (default: false)*

**aria2c \*.torrent –-bt-prioritize-piece=head[=<SIZE>], tail[=<SIZE>]** *#Try to download first and last pieces of each file first. This is useful for previewing files. If head=<SIZE> is specified, pieces in the range of first SIZE bytes of each file get higher priority. tail=<SIZE> means the range of last SIZE bytes of each file. SIZE can include K or M. If size is omitted, SIZE=1M is used.*

**aria2c \*.torrent –-bt-remove-unselected-file=[true|false]** *#Removes the unselected files when download is completed in BitTorrent. To select files, use --select-file option. If it is not specified, all files are assumed to be selected. (default: false)*

**aria2c \*.torrent –-bt-require-crypto=[true|false]** *#Do not accept and establish connection with legacy BitTorrent handshake(19BitTorrent protocol) and use Obfuscation handshake. (default: false)*

**aria2c \*.torrent –-bt-request-peer-speed-limit=<SPEED>** *#If the whole download speed of every torrent is lower than SPEED, it temporarily increases the number of peers to try for more download speed. K or M can be append. (default: 50K)*

**aria2c \*.torrent –-bt-save-metadata=[true|false]** *#Save metadata as “.torrent” file if BitTorrent Magnet URI is used. If the same file already exists, metadata will not be saved. (default: false)*

**aria2c \*.torrent –-bt-stop-timeouot=<SEC>** *#Stop BitTorrent download if download speed is 0 in consecutive SEC seconds. 0 is default and it means this option is disabled.*

**aria2c \*.torrent –-bt-tracker=<URI>[,… ]** *#Comma separated list of additional BitTorrent tracker’s announce URI. It doesn’t affected by --bt-exclude-tracker.*

**aria2c \*.torrent –-bt-tracker-connect-timeout=<SEC>** *#Set the connect timeout in seconds to establish connection to tracker. It works for first time connection only, then --bt-tracker-timeout option is used instead. (default: 60)*

**aria2c \*.torrent –-bt-tracker-timeout=<SEC>** *#Set timeout in seconds. (default: 60)*

**aria2c \*.torrent –-follow-torrent=[true|false|mem]** *#true: download both .torrent file and all files mentioned in it. mem: keep .torrent in memory until files have been downloaded which mentioned in it. false: only download the .torrent file, not the files inside it. (default: true)*

**aria2c \*.torrent -u|–-max-upload-limit=<SPEED>** *#Set max upload speed per each torrent in bytes/sec. 0 is default and means unlimited. K or M can be append to determine speed in K or M*

**aria2c \*.torrent –-max-overall-upload-limit=<SPEED>** *#Set max overlall upload speed in bytes/sec. 0 is default and means unlimited. K or M can be append to determine speed in K or M*

**aria2c \*.torrent –-peer-agent=<PEER\_AGENT>** *#Specify the string used during the BitTorrent extended handshake for the peer’s client version.*

**aria2c \*.torrent –-peer-id-prefix=<PEER\_ID\_PREFIX>** *#Specify the prefix of peer ID. The peer ID in BitTorrent is 20 byte length. If more than 20 bytes are specified, only first 20 bytes are used. If less than 20 bytes are specified, random byte data are added to make its length 20 bytes.*

**aria2c \*.torrent –-seed-time=<MINUTES>** *#Specify seeding time in (fractional) minutes. 0 disables seeding after download completed.*

**aria2c \*.torrent –-seed-ratio=<RATIO>** *#Specify share ratio. Seed completed torrents until share ratio reaches RATIO. Specify 0.0 if the intention is to do seeding regardless of share ratio. If --seed-time option is specified along with this option, seeding ends when one condition is satisfied. (default: 1.0)*

**Aria2c Metalink Specific Options**

**aria2c -M|--metalink-file=\*.meta4** *#The file path to .meta4 and .metalink file. Reads input from stdin when – is specified. This option is not required to specify the metalink file.*

**aria2c \*.meta4 –-follow-metalink=[true|false|mem]** *#true: download both .meta4 file and all files mentioned in it. mem: keep .meta4 in memory until files have been downloaded which mentioned in it. false: only download the .meta4 file, not the files inside it. (default: true)*

**aria2c \*.meta4 –-metalink-language=<LANGUAGE>** *#The language of the file to download.*

**aria2c \*.meta4 –-metalink-location=<LOCATION>[,… ]** *#The location of the preferred server. A comma-delimited list of location is acceptable.*

**aria2c \*.meta4 –-metalink-os=<OS>** *#The operating system of the file to download.*

**aria2c \*.meta4 –-metalink-version=<VERSION>** *#The version of the file to download.*

**aria2c \*.meta4 –-metalink-preferred-protocol=<http|https|ftp|none>** *#Specify preferred protocol. (default: none)*

**Aria2c Additional Options**

**aria2c uri --allow-overwrite=[true|false]** *#Restart download from scratch if the corresponding control file doesn’t exist.*

**aria2c uri --conditional-get=[true|false]** *#Download file only when the local file is older than remote file. This function works only with HTTP(S). To overwrite existing file, --allow-overwrite is required. (default: false)*

**aria2c uri --console-log-level=[debug|info|notice|warn|error]** *#Set log level output to console. (default: notice)*

**aria2c uri --log-level=[debug|info|notice|warn|error]** *#Set log level output. (default: debug)*

**aria2c uri -D|--daemon=[true|false]** *#Run as daemon. The current working dir will be changed to / and standard input, standard output and standard error will be redirected to /dev/null. (default: false)*

**aria2c uri –-disk-cache=<SIZE>** *#Enable disk cache. 0 means caching is disabled. This cache storage is created and shared for all downloads. This feature caches the downloaded data in memory, which grows to at most SIZE bytes. It reduces disk I/O because data will be written on disk in larger unit and it is reordered by the offset of the file. K or M can be used. (default: 16M)*

**aria2c \*.torrent –-on-bt-download-complete=<COMMAND>** *#Set the command to be executed after download completed of BitTorrent. Event Hook method is applied for command*

**aria2c uri –-on-download-complete=<COMMAND>** *#Set the command to be executed after download completed. Event Hook method is applied for command*

**aria2c uri –-on-download-error=<COMMAND>**

**aria2c uri –-on-download-pause=<COMMAND>**

**aria2c uri –-on-download-start=<COMMAND>**

**aria2c uri –-on-download-stop=<COMMAND>**

*#Event Hook*

*#It is possible to specify arbitrary command after specific event has occurred. 3 arguments will pass to specified command when it is executed. Args are; GID, number of files, file path.*

*Example:*

*$ cat hook.sh*

#!/bin/sh

echo “Called with [$1] [$2] [$3]”

$ aria2c –on-download-complete hook.sh http://example.org/file.iso

Called with [1] [1] [/path/to/file.iso]

**aria2c uri –-optimize-concurrent-downloads=[true|false|<A>:<B>]** *#Optimizes the number of concurrent downloads according to the bandwidth available. Adaptation of number of downloads launched in parallel is based on speed observed in the previous downloads. Num of parallel downloads is calculated as N=A+BLog10(Mbps). The default values (A=5, B=25) lead to using typically 5 parallel downloads on 1Mbps networks and above 50 on 100Mbps networks. (default: false)*

**aria2c uri –-summary-interval=<SEC>** *#Set interval in seconds to output download progress summary.0 suppresses the output. (default: 60)*

**aria2c uri -Z|–-force-sequential=[true|false]** *#Fetch URIs in the command-line sequentially and download each URI in a separate session, like the usual command-line download utilities. (default: false)*

**aria2c uri –-max-overall-download-limit=<SPEED>** *#Set max overall download speed in bytes/sec. 0 which is default, means unlimited. K or M can be applied to specify 1024 bytes or 1024K.*

**aria2c uri –-max-download-limit=<SPEED>** *#Set max download speed per each download in bytes/sec. 0 which is default, means unlimited. K or M can be applied to specify 1024 bytes or 1024K.*

**aria2c uri -P|–-parameterized-uri=[true|false]** *#Enable parameterized URI support. Set of parts can be specified as: http://{sv1,sv2,sv3}/foo.iso. Number of sequences also can be specified even with counter: http://host/image[000-100:2].img. Step counter is optional. (default: false)*

**aria2c uri -q|–-quiet=[true|false]** *#No console output. (default: false)*

**aria2c uri –-remove-control-file=[true|false]** *#Remove control file before download. Using with –allow-overwrite=true, download always starts from scratch. This will be useful for users behind proxy server which disables resume.*

**aria2c uri –-save-session=<FILE>** *#Save error/unfinished downloads to FILE on exit. You can pass this output file aria2c with –input-file option on restart. If you like the output to be gzipped append a .gz extension to the file name.*

**aria2c uri –-stop-with-process=<PID>** *#Stop application when process PID is not running. This is useful if aria2 process is forked from a parent process. The parent process can fork aria2 with its own pid and when parent process exists for some reason, aria2 can detect it and shutdown itself.*

**Docker Cheat Sheet. Compiled by Sisay Zeleke tosisay@gmail.com**

**Dockerfile**

**ADD file1.txt file2.txt \*.png /usr/src/data/** *#Copies new files or directories from <src> and adds them to the filesystem of the image at the path <dest>. In this case multiple source files are specified. If local src file is gzip, bzip2,xz, the archive will be extracted in dest.*

**ADD https://example.com/archive.zip /usr/src/data/** *#Unlike COPY, ADD can fetch from http(s).*

**ADD https://github.com/moby/buildkit.git#v0.10.1 /buildkit** *#Adds (clones) source (github) contents to the image destination (buildkit) directory.*

**FROM busybox:latest AS newBuildName** *#Initializes a new build stage and sets the base image for subsequent instructions. Optionally a name can be given to a new build stage by adding AS name, which can be used in subsequent ‘from’ instructions to refer to the image built in this stage.*

**RUN apt-get update && apt-get install -y curl** *#RUN executes any commands to create a new layer on top of the current image.*

**RUN –-mount=type=bind,source=/host/hdata,target=/container/cdata,rw=true** *#--mount allows to mount files or directories from host system into the container during build process as read-only by default. target|dst|destination are interchangeable to determine mount path. Contents of hdata will be found inside cdata. Bind mounts allow to create direct link between host and container fs in which changes in one are immediately reflected in the other.*

**RUN –-mount=type=bind,from=newBuildName,source=/subPathFromBuildStage/hdata,target=/container/cdata,readwrite=true** *#Mounts files or directories from build stage. Source directory/file should be subpath in the ‘from’ stage.*

**RUN –-mount=type=cache,target=/var/cache/apt,sharing=locked,readonly=true** *#Cache is used for reusable data but can be discarded at any time. Cache mount type allows the build container to cache directories for compilers and package managers so that every command starts with apt will be cached and it uses for better performance. sharing=shared: for multiple writers, private: to create new mount if there are multiple writers, locked: pauses the second writer until the first one releases the mount in this case, apt needs exclusive access to its data. mode for file mode, uid for user id and gid for group id can be specified. (default: mode=0755,uid=0,gid=0)*

**RUN –mount=type=tmpfs,target=/var/tmp,size=100M** *#Tmpfs is in-memory fs inside container and doesn’t persist if container is stoped. Allows mounting tmpfs in the build container with mount destination and its max size of the fs.*

**RUN --network=none pip install --find-links myPackageDir myPackage** *#--network allows control over which networking environment the command is run in. ‘none’: runs with no network access and, ‘host’: run in the host’s network environment. In this case, pip will only be able to install the packages provided in the tar file (myPackage) which can be controlled by an earlier stage.*

**CMD [“echo”, “Hello”, “World”]** *#CMD instruction sets the command to be executed when running a container from an image. “echo”: executable or command, “Hello”: param1, “World”: param2*

**ENTRYPOINT [“echo”]**

**CMD [“Hello”, “World”]** *#If the container is going to run the same executable every time, then the executable can be specified by ENTRYPOINT*

**CMD echo Hello World** *#This form runs the command in a shell (/bin/sh -c), and assumed it is typed inside in terminal of the container.*

**EXPOSE 80/tcp** *#Informs Docker that the container listens on the specified network ports at runtime. It is only to tell the user which port is ready for publish and should be published at the time of executing docker run command.*

**ENV MY\_NAME=”John Doe”**

**ENV MY\_DOG=Rex\ The\ Dog**

**ENV MY\_CAT=fluffy** *#Sets environment variable using <key>=<value> combination. These values will be in the environment for all subsequent instructions in the build stage. Multiple key=value entries can be set with a single ENV instruction and equal sign can be omitted for a single entry.*

**COPY file1.txt file2.txt \*.png /usr/src/data/** *#Copies new files or directories from build context <src> and adds them to the filesystem of the image at the path <dest>. In this case multiple source files are specified.*

**COPY --from=newBuildName /myapp /usr/bin/** *#Files and directories can be copied from the build context, build stage, named context, or an image.*

**COPY -–chown=myuser:mygroup --chmod=644 files\* /somedir/** *#Copy files or directories with specified owner and access mode. Username and groupname can be replaced by userId and groupId respectively. Group information is optional. This command is supported with only linux containers and if the container root filesystem doesn’t contain either /etc/passwd or /etc/group files and either user or group names are doesn’t exist, the build will fail on the COPY.*

**COPY -–exclude=\*.jpg –-exclude=\*.md files\* /somedir/** *#Exclude some files with specified format when copying.*

**VOLUME /myvol** *#Creates a mount point with the specified name and marks it as holding externally mounted volumes from native host or other containers.*

**USER UID[:GID]** *#Sets the user name (or UID) and optionally the user group (or GID) to use as the default user and group for the reminder of the current stage. The specified user is used for RUN instructions and at runtime, runs the relevant ENTRYPOINT and CMD commands.*

**WORKDIR /dirFromRoot** OR

**WORKDIR dirFromCurrentWD** *#Sets the working directory for any instructions that follow it in the Dockerfile. If the WORKDIR doesn’t exist, it will be created.*

**ENV DIRPATH=/path**

**WORKDIR $DIRPATH/DIRNAME** *#Even it can be resolve environment variables previously set using ENV. Environment variables should explicitly set in Dockerfile.*

**ARG user1** OR

**ARG user1=Peter** *#Defines a variable that users can pass at build-time to the builder with the ‘docker build’ command using the ‘-–build-arg <varname>=<value>’. Values assigned for the variable is assumed as default value. The variable will go out of scope at the end of the build stage where it was defined.*

**HTTP\_PROXY, http\_proxy, HTTPS\_PROXY, https\_proxy, FTP\_PROXY, ftp\_proxy, NO\_PROXY, no\_proxy, ALL\_PROXY, all\_proxy** *#All these are predefined ARGs that can be called using --build-arg without creating them.*

**STOPSIGNAL SIG<NAME>** *#Sets the system call signal that will be sent to the container to exit. Signal can be name or number. The default is SIGTERM and it can be override per container using ‘-—stop-signal’ flag when using ‘docker run’ or ‘docker create’.*

**SHELL [“/bin/sh”, “-c”]** *#Allows the default shell used for the shell form of commands to be overridden by determining executable and parameters as specified. So that any consecutive commands are executed based on specified shell.*

**RUN <<EOT bash**

**set -ex**

**apt-get update**

**apt-get install -y vim**

**EOT** *#Here-Documents allow redirection of subsequent Dockerfile lines to the input of RUN or COPY commands.*

**Compose file: Name top-level element**

**name: myapp** *#Defines the project name. It exposed for environment variable resolution as ‘COMPOSE\_PROJECT\_NAME’, issuing ‘command: echo “I’m running ${COMPOSE\_PROJECT\_NAME}” displays the project name in container.*

**Compose file: Services top-level elements**

**services:** *#A service is an abstract definition of a computing resource within an application and defines runtime constraints.*

**web:** *#A service name*

**image: nginx:latest** *#An image name to be used in this service*

**image: redis** *#Image name must follow the Open Container Specification and all these names are valid*

**image: redis:5**

**image: redis@sha256:4tydwjk9pmcew4635yf354ewch686u54**

**image: library/redis**

**image: docker.io/library/redis**

**image: my\_private.registry:5000/redis**

**container\_name: my-web-container** *#Specifies a custom container name*

**platform: darwin** *#Defines the target platform the containers for the service run on. It uses the os[/arch[/variant]] syntax*

**platform: windows/amd64**

**platform: linux/arm64/v8**

**profiles: [“frontend”]** *#Defines a list of named profiles for the service to be enabled under. If assigned, service will be started only if profile is activated*

**profiles:** *#Using list:*

**- debug**

**pull\_policy: missing** *#Defines the decisions Compose makes when it starts to pull images. Missing is default (pulls only if it’s not available in the platform cache), always: always pulls the image from the registry, never: don’t pull the image from registry and use cached image only, if\_not\_present: considered an alias for this value for backward compatibility, build: rebuilds the image if it’s already present*

**read\_only: true** *#Configure the service container to be created with a read-only filesystem*

**restart: “no”** *#Defines the policy that the platform applies on container termination. “no”: Does not restart the container under any circumstances (default policy)*

**restart: always** *#Restarts the container until its removal*

**restart: on-failure:3** *#Restarts the container if the exit code indicates an error, optionally limit the number of restart*

**restart: unless-stopped** *#Restarts the container irrespective of the exit code but stops restarting when service is stopped|removed*

**build:** *#Specify build configuration for creating a container image from source.*

**context: ./config\_dir** *#Directory for build configuration. Consecutive command access based on this location*

**dockerfile: Dockerfile.nginx** *#File name for the Dockerfile. If the file name is ‘Dockerfile’, it is not necessarily to specify here since it is default name.*

**stdin\_open: true** *#Configures a service’s container to run with an allocated stdin. Same as running container with -i flag*

**tty: true** *#Configures a service’s container to run with a TTY. Same as running container with -t or –tty flag*

**stop\_grace\_period: 1m30s** *#Specifies how long Compose must wait when attempting to stop a container if it doesn’t handle SIGTERM (default for stop\_signal) before sending SIGKILL. (default: 10s)*

**stop\_signal: SIGUSR1** *#Defines the signal that Compose uses to stop the service containers*

**init: true** *#Runs an init process(PID 1) inside container that forwards signals and reaps processes*

**working\_dir: /usr/src/app** *#Overrides the container’s working directory which is specified by the image (Dockerfile’s WORKDIR)*

**depends\_on:** *#Used to control the order of service startup and shutdown*

**- backend** *#Service name*

OR

**depends\_on:**

**db:** *#Service name*

**condition: service\_started** *#An equivalent of the short syntax described above*

**restart: true** *#Restarts this service after it updates the dependency service.*

**required: false** *#Only warns if the dependency service isn’t started or available*

**redis:**

**condition: service\_healthy** *#Specifies that a dependency is expected to be “healthy” as indicated by healthcheck before starting a dependent service*

**policy:**

**condition: service\_completed\_successfully** *#Specifies that a dependency is expected to run to successful completion before starting a dependent service*

**healthcheck:** *#Declares a check that’s run to determine whether or not the service containers are “health”.*

**test: [“CMD”, “curl”, “-f”, “http://localhost”]** *#Test defines the command Compose runs to check container health. In this condition it hit the local web app (localhost)*

**interval: 1m30s**

**timeout: 10s**

**retries: 3**

**start\_period: 40s**

**start\_interval: 5s**

**healthcheck:** *#Health check set by the image can be disabled from Compose*

**disable: true**

**hostname: my\_host** *#Declares a custom host name to use for the service container*

**entrypoint: /code/entrypoint.sh** *#Declares the default entrypoint for the service container*

**entrypoint:** *#Alternatively the value can also be a list*

**- php**

**- -d**

**- zend\_extension=/usr/local/lib/php/exts/xdebug.so**

**- -d**

**- memory\_limit=-1**

**- vendor/bin/phpunit**

**command: bundle exec thin -p 3000** OR

**command: [ “bundle”, “exec”, “thin”, “-p”, “3000” ]** *#overrides the default command declared by the container image, for example by Dockerfile’s CMD*

**Compose file: Volume top-level elements**

**services:**

**service\_name:**

**volumes:** *#Used to define mount host files*

**- type: bind** *#Creates direct link with host’s file. So that changes in one location will be applied on the other. (other types are volume, tmpfs, npipe, cluster)*

**source: ./nginx.conf** *#Mounts config file from the host into container*

**target: /etc/nginx/default.conf** *#Mount location inside container*

**read\_only: true** *#Flags the volume as read-only*

**bind:** *#Used to configure additional bind options*

**propagation: true** *#Propagation mode used for the bind*

**create\_host\_path: true** *#Creates a directory at the source path on host if there is nothing present*

**volume:** *#Configures additional volume options*

**nocopy: true** *#Flag to disable copying of data from a container when a volume is created*

**subpath: sub** *#Path inside a volume to mount instead of the volume root*

**tmpfs:** *#Configures additional tmpfs options*

**size: 500m** *#Size for the tmpfs mount in bytes|numeric*

**mode: 755** *#File mode for the tmpfs as Unix permission bits as an octal number*

**volumes\_from:** *#Mounts all of the volumes from another service or container. ro|rw can be specified*

**- service\_name**

**- service\_name:ro**

**- container:container\_name**

**- container:container\_name:rw**

**volumes:** *#To use a volume across multiple services, each service must granted access by using top-level volume element*

**db-data:** *#Usedtup and shutdown*

**driver: foobar** *#Specifies which volume driver should be used. It must exist before Compose runs*

**external: true** *#Specifies this volume must be existed on the platform and its lifecycle is managed outside of that of the application*

**name: “my-app-data”** *#Sets custom name for a volume*

**name: ${DATABASE\_VOLUME}** *#It is possible this lookup name a parameter of the Compose file. In this case, DATABASE\_VOLUME=volume\_001 is in .env file*

**labels:** *#Used to add metadata to volumes as an array or dictionary*

**com.example.description: “Database volume”**

**- “com.example.department=IT/Ops”**

**driver\_opts:** *#Specifies a list of options as key-value pairs to pass to the driver for this volume. Options are driver-dependent*

**type: “nfs”**

**o: “addr=10.40.0.199,nolock,soft,rw”**

**device: “:/docker/example”**

**Compose file: Access Control elements**

**services:**

**service\_name:**

**privileged: true** *#run the container with elevated privileges*

**user: ubuntu** *#Overrides the user used to run container process. (default: root, unless set by Dockerfile USER)*

**group\_add:** *#Specifies additional groups by name or number which the user inside the container must be a member of. Useful for reading/writing on a shared volume which requires specific group who owns the file*

**- mail**

**cap\_add:** *#Specifies additional container capabilities as strings*

**- ALL**

**cap\_drop:** *#Specifies container capabilities to drop as strings*

**- NET\_ADMIN**

**- SYS\_ADMIN**

**Compose file: Networks top-level elements**

**services:**

**service\_name:**

**mac\_address: a1:b2:c3:d4:e5:f6** *#Sets a MAC address for the service container. If container runtime rejects it, use “networks.mac\_address” instead*

**extra\_hosts:** *#Adds hostname mappings to the container network interface configuration (/etc/hosts for linux). Separator = and : are interchangeable*

**- “somehost=162.242.195.82”**

**- “otherhost1=50.31.209.229”**

**- “otherhost2:50.31.209.229”**

**- “myhostv6=::1”**

**- “anotherhostv6:::1”**

**extra\_hosts:** *#Or using long syntax:*

**somehost: “162.242.195.82”**

**otherhost: “50.31.209.229”**

**myhostv6: ”::1”**

**expose:** *#Defines the incoming port or a range of ports that Compose exposes from the container. If protocol is not specified, TCP will be used*

**- “3000”**

**- “8000”**

**- “8080-8085/tcp”**

**ports:** *#Port numbers to expose for this service*

**- “8080:80”**

**dns: 8.8.8.8** OR

**dns:**

**- 8.8.8.8**

**- 4.4.4.4** *#Defines custom DNS servers to set on the container network interface configuration in the form of single value or a list*

**network\_mode: “host”** *#Sets a service container’s network mode. “host” gives the container raw access to the host’s network interface*

**network\_mode: “none”** *#Turns off all container networking*

**network\_mode: “service:[service name]”** *#Gives the containers access to the specified service only*

**networks:** *#Defines the networks that service containers are attached to, referencing entries under the ‘networks’ top-level element.*

**- some-network**

**- other-network**

**networks:** *#Specifying static IP addr for service container*

**some-network**

**ipv4\_address: 172.16.238.10**

**networks:**

**some-network**

**name: my-app-net** *#Set custom name for the network. If the network name contains special character, this name is used as a reference*

**name: “${NETWORK\_ID}”** *#This can also possible and Compose file doesn’t need to hard-code runtime specific values*

**priority: 1000** *#Priority indicates in which order Compose connects the service’s containers to its networks. In this case, some-network has highest priority. (Default: 0)*

**internal: true** *#By default, Compose provides external connectivity to networks but this option is used to isolate this network from external networks*

**proxy-network**

**external: true** *#Lifecycle of this network is maintained outside of that of the application. Compose doesn’t attempt to create this network so it should be existed before running*

**ports:** *#Used to define the port mappings between the host machine and the containers.*

**- “3000”**

**- “3000-3005:8080-8081”**

**- “127.0.0.1:5000-5010:5000-5010”**

**- “6070:6070/udp”**

**ports:** *#For more human-readable and for documentation purpose:*

**- name: web** *#Port name for readable purpose*

**target: 80** *#The container port*

**host\_ip: 127.0.0.1** *#Host IP mapping*

**published: “8080”** *#Publicly exposed port*

**protocol: tcp** *#Port protocol*

**app\_protocol: http** *#Application protocol this port is used for*

**mode: host** *#host: for publishing a host port on each node, ingress: for a port to be load balanced(default)*

**networks:** *#Top level section to define subnet for static IPs*

**some-network**

**labels:** *#Add metadata to containers using array or dictionary*

**com.example.description: “Financial transaction network”**

**- “com.example.department=Finance”**

**ipam:**

**attachable: true** *#Standalone containers should be able to attach to this network. Containers attached to this network can communicate each other*

**driver: default**

**config:**

**- subnet: 172.28.0.0/16**

**ip\_range: 172.28.5.0/24**

**gateway: 172.28.5.254**

**aux\_addresses:** *#Auxiliary addresses used by Network driver as a mapping from hostname to IP*

**host1: 172.28.1.5**

**host2: 172.28.1.6**

**host3: 172.28.1.7**

**options:** *#Driver-specific options as a key-value mapping*

**foo: bar**

**baz: “0”**

**driver\_opts:** *#Specifies a list of options as key-value pairs to pass to the driver*

**com.docker.network.bridge.host\_binding\_ipv4: “127.0.0.1”**

**Compose file: Resource elements**

**services:**

**service\_name:**

**storage\_opt:** *#Defines storage driver options for a service*

**size: ‘1G’**

**tmpfs:** *#Mounts a temporary file system inside the container. Can be single value or a list*

**- /run**

**- /tmp**

**mem\_limit: “2147483648”** *#Configures a limit on the amount of memory a container can allocate, set as a string expressing a byte value; 2GB in this case*

**mem\_reservation: “1073741824”** *#Configures a reservation on the amount of memory a container can allocate, set as a string expressing a byte value; 1GB in this case*

**mem\_swappiness: “50”** *#Defines as a percentage, a value between 0 – 100 for the host kernel to swap out anonymous memory pages used by a container*

**memswap\_limit: “1g”** *#Defines the amount of memory the container is allowed to swap to disk. If “memory” is set to “300m”, the swap size will be 700m in this case. If it is set to -1, the container is allowed to use unlimited swap*

**shm\_size: 64m** *#Configures the size of the shared memory (/dev/shm partition) allowed by the service container. Specified as a byte\_value*

**ipc: “shareable”** *#Configures the IPC isolation mode set by the service container. Shareable gives the container its own private IPC namespace with a possibility to share it with other containers.*

**ipc: “service:[service name]”** *#service name makes the container join another container’s shareable IPC namespace*

**com.example.foo: bar** *#Map type annotation*

**- com.example.foo=bar** *#Array type annotation*

**devices:**

**- “/dev/ttyUSB0:/dev/ttyUSB0”**

**- “/dev/sda:/dev/xvda:rwm”** *#Defines a list of device mappings for created containers with cgroup permissions*

**blkio\_config:** *#Defines a set of configuration options to set block IO limits for a service*

**weight: 300** *#Modify the proportion of bandwidth allocated to a service relative to other services. (values: 10 – 1000, default: 500)*

**weight\_device:** *#Fine-tune bandwidth allocation by device*

**- path: /dev/sda** *#Defines the symbolic path to the affected device*

**weight: 400** *#An integer value between 10 - 1000*

**device\_read\_bps:** *#Set a limit in bytes per second for read operation on a given device*

**- path: /dev/sdb**

**rate: ‘12mb’** *#Either as an integer value representing the number of bytes or as a string expressing a byte value*

**device\_read\_iops:** *#Set limit in operations per second for read operations on a given device*

**- path: /dev/sdb**

**rate: 120** *#As an integer value representing the permitted number of operations per second*

**device\_write\_bps:** *#Set a limit in bytes per second for write operation on a given device*

**- path: /dev/sdb**

**rate: ‘1024k’**

**device\_write\_iops:** *#Set limit in operations per second for write operations on a given device*

**- path: /dev/sdb**

**rate: 30**

**cpu\_count: 4** *#Defines the number of usable CPUs for service container*

**cpu\_percent: 50%** *#Defines the usable percentage of the available CPUs*

**cpu\_shares: 2** *#Defines as integer value a service container’s relative CPU weight versus other containers*

**cpu\_period:** *#Configures CPU CFS (Completely Fair Scheduler) period when a platform is based on Linux kernel*

**cpu\_quota:** *#Configures CPU CFS quota when a platform is based on Linux kernel*

**cpu\_rt\_runtime: ‘400ms’** OR **95000`** *#Configures CPU allocation parameters for platforms with support for realtime scheduler. It can be either an integer value using microseconds as unit or a duration*

**cpu\_rt\_period: ‘1400us’** OR **11000`**

**cpus: 1.5** *#Define the number of (potentially virtual) CPUs to allocate to service containers. This is a fractional number. 0.000 means no limit*

**cpuset: 0-2** *#Defines the explicit CPUs in which to allow execution. Can be a range 0-3 or a list 0,1*

**Compose file: Configuration top-level elements**

**services:**

**service\_name:**

**labels:** *#Add metadata to containers using either an array or map. Reverse-DNS notation is recommended*

**com.example.description: “Accounting webapp”**

**com.example.department: “Finance”**

**com.example.label-with-empty-value: “”**

**labels:** *#Array format*

**- “com.example.description=Accounting webapp”**

**- “com.example.department=Finance”**

**- “com.example.label-with-empty-value”**

**logging:** *#Defines the logging configuration for the service*

**driver: syslog** *#Logging driver for the service’s containers*

**options:**

**syslog-address: “tcp://192.168.0.42:123”**

**annotations:** *#Attach metadata to a resource. Specifically describe OCI image components such as manifests, indexes and descriptors*

**extends:** *#Shares common configurations among different files, or even different projects entirely*

**file: common.yml** *#This compose file is optional*

**service: webapp** *#Refer to another Compose file (common.yml) and select this service to deploy on this project. Specifying service is required*

**env\_file:** *#Used to specify one or more files that contain environment variables to be passed to the containers*

**- ./a.env**

**- ./b.env** OR

**env\_file: ./myenvs.env**

**env\_file:** *#Can also be declared as a mapping*

**- path: ./default.env**

**required: true**

**- path: ./override.env** OR

**required: false** *#If .env file is missing, Compose silently ignores the entry*

**environment:** *#Defines environment variables set in the container. Can use either an array or a map. Any boolean values (true, false, yes, no) should be enclosed in quotes to ensure they as literal by YAML parser. Variables in environment has higher precedence than in env\_file. The following is map syntax:*

**RACK\_ENV: development**

**SHOW: “true”**

**USER\_INPUT:**

*#The following is array syntax:*

**environment:** *#Can also be declared as a mapping*

**- RACK\_ENV=development**

**- SHOW=true**

**- USER\_INPUT**

**environment:** *#Used to configure basic environment variables*

**WEB\_USER: example\_name**

**WEB\_DB: postgres\_db**

**configs:**

**- my\_config**

**- my\_other\_config**

**configs:** *#Configs allows services to adapt their behavior without the need to rebuild a Docker image. Services can only access configs when explicitly granted by a configs top-level element*

**my\_config:**

**name: my\_config\_forhttp** *#A name to reference the config file*

**name: “${HTTP\_CONFIG\_KEY}”** *#Can be assign name from parameter HTTP\_CONFIG\_KEY*

**file: ./httpd.conf**

**my\_other\_config:**

**external: true** *#Assumed as external file already existed*

**app\_config:**

**content: |** *#Created when the application is deployed, by registering the inline content as the configuration data*

**debug=${DEBUG}**

**spring.application.admin.enabled=${DEBUG}**

**spring.application.name=${COMPOSE\_PROJECT\_NAME}**

**+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++**

**Docker CLI**

**docker run --help** *#To list the help on any command just execute the command, followed by –help option*

**docker buildx build –-load -t|--tag <image-name> .** *#build the image using default builder or context. --load is used to load the image locally*

**docker buildx build --progress=plain --load -t|--tag <image-name> .** *#Used to display debugging information with more verbose output for debugging*

**docker run -i|–-interactive -t|--tty <image-name>** *#To run the image with interactive way. Interactive is used to keep the STDIN(STandarD INput) open. Tty allocates a pseudo-TTY(a terminal) which makes the terminal session interactive and behaves like a regular terminal*

**docker run -–add-host www.example.com=172.16.1.11 -i|–-interactive -t|--tty <image-name>** *#Add hosts into a build container’s /etc/hosts file. Multiple entry is allowed to add additional host information*

**docker run –-add-host host.docker.internal=host-gateway -i|–-interactive -t|--tty <image-name>** *#To connect to services running on the host*

**docker buildx build –-build-arg HTTP\_PROXY=http://10.20.30.2:1234 –-build-arg FTP\_PROXY=http://40.50.60.5:4567 .** *#The –-build-arg flag allows to pass build-time variables to Docker images, making the build process more flexible and customizable. These variables can be defined in the Dockerfile using the ARG instruction and passed when building the image.(ARG HTTP\_PROXY, ARG FTP\_PROXY,….)*

**export HTTP\_PROXY=http://10.20.30.2:1234**

**docker buildx build –-build-arg HTTP\_PROXY .** *#In this case, the daemon propagates the value from the local environment into the Docker container it’s building.*

**--build-context=name=VALUE**

**docker buildx build –-build-context alpine=docker-image://alpine@sha0123456789 .** *#Defines additional build context with specified contents. In Dockerfile the context can be accessed when ‘FROM name’ or ‘—from=name’ is used. When Dockerfile defines a stage with the same name, it is overwritten.*

**docker buildx build –-build-context project=https://github.com/usr/proj.git .** *#Copies proj to specified location in build context: (COPY -–from=project myfile /) from Dockerfile*

**docker buildx build –-cache-from=type=registry,ref=my-registry.com/my-image:cache -–cache-to=type=registry,ref=my-registry.com/my-image:cache,mode=max -t|--tag my-image:latest .** *#Use an external cache source for a build. ‘registry’ can import cache from a cache manifest or special image configuration on the registry. Using a Docker registry such as Docker Hub, GitLab Container Registry, or private registry as the cache source is common for Docker builds. ‘From’ determines previously cached ‘to’ location.*

**docker buildx build –-cache-from=type=local,src=/path/to/local/cache -–cache-to=type=local,dest=/path/to/local/cache,mode=max -t|--tag my-image:latest .** *#For iterative development, on local machine, cache locally on local storage is a good option.*

**docker buildx build –-cache-from=type=s3,src=s3://my-s3-bucket/docker-build-cache -–cache-to=type=s3,dest=s3://my-s3-bucket/docker-build-cache,mode=max -t|--tag my-image:latest .** *#Using Amazon S3 as a cache source, configure the aws CLI, aws credentials and set permissions to access the S3 bucket, and specify the S3 bucket location in the cache configuration.*

**docker buildx build –-cache-from=my-image:latest –-cache-to=type=inline -t|--tag my-image:latest .** *#Writes the cache metadata into the image configuration. This approach is helpful for self-contained builds, where you want the cache to be included within the image itself rather than in an external cache source.*

**docker buildx build –f|--file <filepath> .** *#Specifies the filepath of the Dockerfile to use. If unspecified, a file named ‘Dockerfile’ at the root of the build context is used by default.*

**cat Dockerfile | docker buildx build –f|--file - .** *#To read a Dockerfile from stdin, you can use ‘-’ as the argument for –-file.*

**docker buildx build --load --metadata-file metadata.json .** *#To output build metadata such as the image digest and the metadata will be writtn as a JSON object to the specified file. The directory of the specified file must already exist and be writable. To see the output metadata, use ‘cat metadata.json’ in the current build context directory.*

**docker buildx build --network default|none|host -t|--tag my-image:latest .** *#Set the networking mode for the RUN instructions during build. default: Run in the default network. none: Run with no network access. host: Run in the host’s network environment.*

**docker buildx build –-no-cache-filter stage1,stage2,stage3 -t|--tag my-image:latest .** *#Allows to control which stages in a multi-stage Docker build skip cache. Other stages not listed will still use the cache if available.*

*FROM oven/bun:1 as base*

*…*

*FROM base AS install*

*…*

*FROM base AS release*

*…*

**docker buildx build --no-cache-filter base,release .** *#Ignore the cache ‘base’ and ‘release’ stages.*

**docker buildx build -t|--tag my-image:latest -o|--output=type=docker .** *#--output option allows to specify the output type and location for the resulting build. ‘type=docker’ exports the built image as a standard Docker image and loads it directly into local Docker daemon. This is the default output and images can be seen with ‘docker images’ command. Default storage location is ‘/var/lib/docker’.*

**docker buildx build -o|--output=type=tar,dest=./my-image.tar .** *#This will output the build as a ‘.tar’ archive at the specified destination. This image can be distributed as a file.*

**docker load -i /path/to/your/image.tar** *#Used to load tar image.*

**docker buildx build -t|--tag my-image:latest -o|--output=type=registry .** *#This will push the built image directly to a Docker registry (like Docker Hub, AWS ECR, etc.) as long as you specify the appropriate registry and tag format in ‘-t’.*

**docker buildx build -t|--tag my-image:latest -o|--output=type=oci,dest=./my-oci-image .** *#Outputs the build in OCI image format to a specified local directory. OCI format is useful if you are working with Kubernetes or tools that support this image format.*

**docker buildx build --target=my-target -o|--output=type=local,dest=./output .** *#Outputs the specified build stage ‘my-target’ files as artifacts into the specified destination directory, rather than as a Docker image. This is helpful when you only need build artifacts (e.g., binaries) without creating a Docker image and it is available for immediate use.*

**docker buildx build --platform=linux/arm/v7 .** *#Set the target platform for the build. All ‘FROM’ commands inside the Dockerfile without their own ‘—-platform’ flag will pull base images for this platform and this value will also be the platform of the resulting image. The default platform is the platform of the BuildKit daemon where the build runs. The format has os/arch/variant form, while variant is optional.*

**docker buildx build --progress=auto|plain|tty|rawjson|json -t|--tag my-image:latest .** *#Set type of progress output. auto: It is the default output type. plain: Used to show container output in a plain text format without any formatting or progress bars. rawjson: Output marshals the solve status events from BuildKit to JSON lines. json: Similar with rawjson but formatted in a more human-readable way. This mode is designed to be parsed by an external program. tty: Docker formats the output to be more suitable for interactive terminal displays. You can also use the ‘BUILDKIT\_PROGRESS’ environment variable to set its value. ‘BUILDKIT\_COLORS’ environment variable also used to modify colors of the terminal output.*

**Vi Improved Cheat Sheet. Compiled by Sisay Zeleke tosisay@gmail.com**

**Exiting**

**:q** *#Close file*

**:qa** *#Close all files*

**:w** *#Save file*

**:wq / :X** *#Save and close file*

**ZZ** *#Save and quit*

**:q! / ZQ** *#Quit without checking changes*

*# Upcoming compilations*

*# vim*

*# docker*

*# matplotlib*

*# conda*

*# scikit-learn*

*# seaborn*

*# git*

*# numpy*

*# grep|tee|sed|awk*

*# regex*

*#*

*# playwright*

*# Python built-in modules 1: os, sys, multiprocessing, threading, concurrent.futures, time, datetime, logging, functools, collections, itertools, pickle, json, math, cmath, argparse, subprocess, csv, statistics, random, re, socket, io, urllib, enum, hashlib, base64, configparser, decimal, fractions, zipfile, struct, pdb, traceback, abc, html, http, uuid*

*# Advanced Python built-in modules 2: contextlib, types, abc, asyncio, typing, dataclasses, inspect, gc, profile, cprofile, timeit, unittest, doctest, ast, array, struct,*

*# Linux: memory, process, services, file system, disk, kernel, security, network, logger, boot loader, devices*