Below we will analyze a portfolio consisting of of Apple(AAPL), Amazon(AMZN), IBM and CISCO. The goal of this analysis if to find the most efficient Allocations(Weight) for each that will provide us the best sharp ratio. We will do this by first evaluating the behavior of the portfolio with arbitrary weights. We will then use Python to create 3000 that will provide us the best Sharp Ratio for this Portfolio.

In [1]:
```
import pandas as pd
import seaborn as sns
from pandas_datareader import data as wb
import matplotlib.pyplot as plt
```

In [2]:
```
from plotly.offline import init_notebook_mode, download_plotlyjs, iplot,
plot
import cufflinks as cf
```

In [3]:
```
init_notebook_mode(connected=False)
cf.go_offline()
```

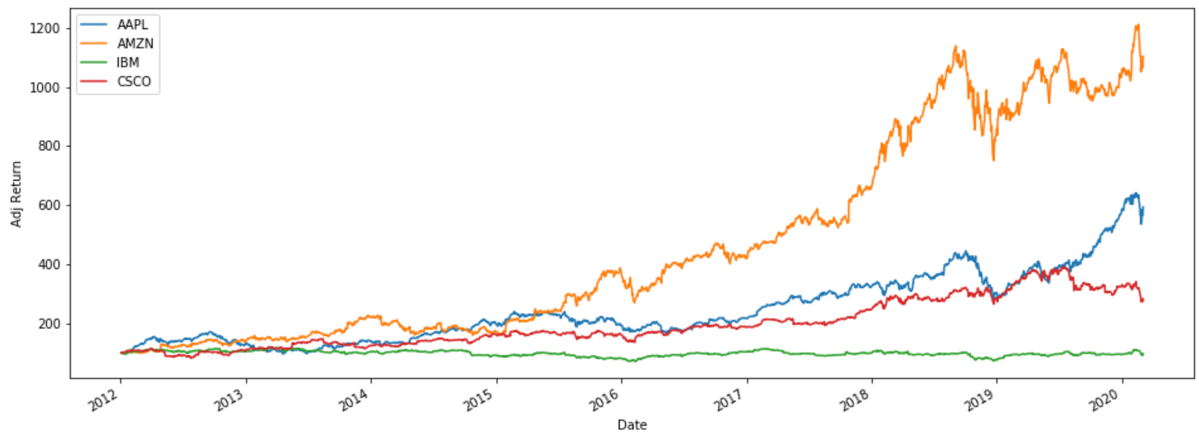
In [4]:
```
tickers = ["AAPL", "AMZN", "IBM", "CSCO"]
```

In [5]:
```
portt = pd.DataFrame()
for t in tickers:
    portt[t] = wb.DataReader(t, data_source="yahoo", start="2012-1-1")[
"Adj Close"]
```

Below we normalize the data to see the movement of the Adjusted Close for each Stock. WE can see that Amazon was the leadnig stock in the portfolio since mid 2015. Apple and Cisco battles fro mearly 2019 to late 2019 where Apple eventually takes the lead. IBM has stayed pretty steady with minimal Volitility over the period.

In [6]:
```
(portt/portt.iloc[0] * 100).plot(figsize = (16,6))
plt.ylabel("Adj Return")
```

Out[6]: Text(0, 0.5, 'Adj Return')



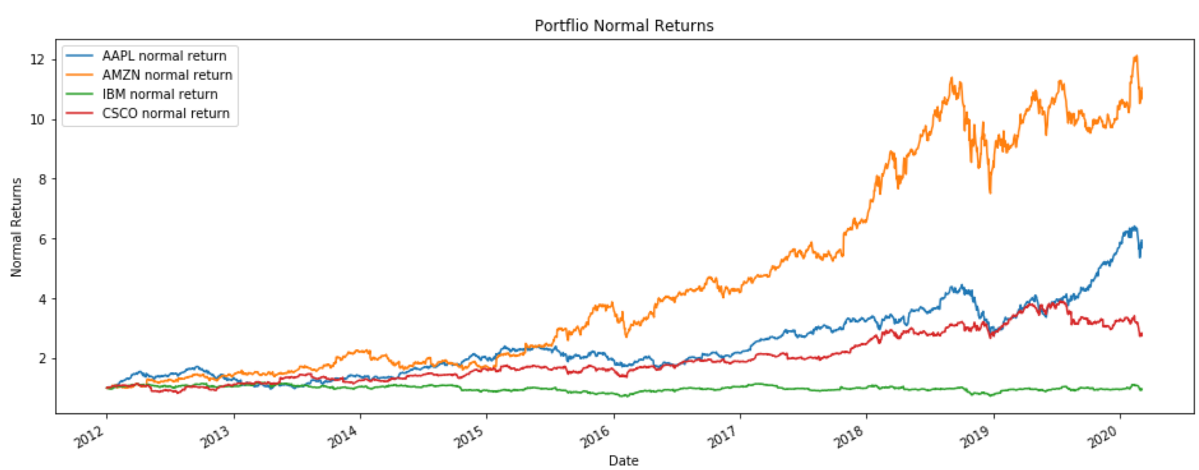
In [ ]:

Calculate the cumulative/normalized returns for each stock in the portfolio. Amazon would have generated the best cumulative returns for thi portfolio at the highest 12+%. This does bring up the question of, would this have been the same if we weighed the portfolio differently?

In [7]:
```
for stock in portt:
    portt[f"{stock} normal return"] = portt[stock]/portt[stock].iloc[0]
```

In [8]:
```
portt[['AAPL normal return','AMZN normal return', 'IBM normal return',
'CSCO normal return']].plot(figsize = (16,6))
plt.ylabel("Normal Returns")
plt.title("Portflio Normal Returns")
```

Out[8]: Text(0.5, 1.0, 'Portflio Normal Returns')



In [9]:
```
portt.head()
```

Out[9]:

Date	AAPL	AMZN	IBM	CSCO	AAPL normal return	AMZN normal return	IBM normal return	CSCO normal return
2012-01-03	50.994907	179.029999	139.934006	14.633397	1.000000	1.000000	1.000000	1.000000
2012-01-04	51.268970	177.509995	139.363144	14.916168	1.005374	0.991510	0.995920	1.019324
2012-01-05	51.838169	177.610001	138.702209	14.861184	1.016536	0.992068	0.991197	1.015566
2012-01-06	52.380054	182.610001	137.109772	14.806202	1.027162	1.019997	0.979817	1.011809
2012-01-09	52.296970	178.559998	136.396225	14.900459	1.025533	0.997375	0.974718	1.018250

In [ ]:

lets create weight allocation for each company and to better understand the performance of each company in the portfolio. Again we will use an arbitrary list of weights to test this approach and see which stock would give us the better return at the set weight

In [10]:
```
## 30% In Apple
## 40 % in Amazon
## 10% in IBM
## 20% in Cisco
weights = [0.3,0.4,0.1,0.2]
for st, allocation in zip(portt,weights):
    portt[f"{st} Allocation"] = portt[f"{st} normal return"] * allocatio
n
```

In [11]:
```
portt.head()
```

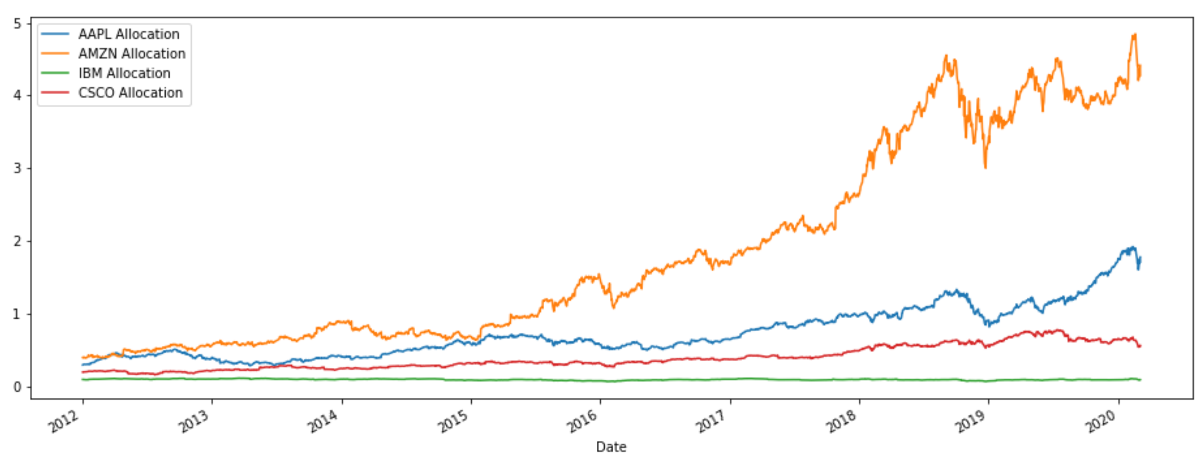
Out[11]:

Date	AAPL	AMZN	IBM	CSCO	AAPL normal return	AMZN normal return	IBM normal return	CSCO normal return	All
2012-01-03	50.994907	179.029999	139.934006	14.633397	1.000000	1.000000	1.000000	1.000000	0
2012-01-04	51.268970	177.509995	139.363144	14.916168	1.005374	0.991510	0.995920	1.019324	0
2012-01-05	51.838169	177.610001	138.702209	14.861184	1.016536	0.992068	0.991197	1.015566	0
2012-01-06	52.380054	182.610001	137.109772	14.806202	1.027162	1.019997	0.979817	1.011809	0
2012-01-09	52.296970	178.559998	136.396225	14.900459	1.025533	0.997375	0.974718	1.018250	0

WE see that Amazon would have made up close to 50% of our portfolio at the end of this period with a 12% return that we discovered above. Below We will calculate our positon value based on a value invested that will be associated the allocations that we assigned above

In [12]:
```
portt[['AAPL Allocation', 'AMZN Allocation', 'IBM Allocation','CSCO Allo
cation']].plot(figsize = (16,6))
```

Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x1c19fe2b38>



WE will calculate the Positon Value of each stock in this portfolio to better understand the movement of our portfolio. We will assume we have invested 100K into a portfolio allocated with the weights we assigned above to understand out position value. We will simply take the Allocation value and multiply it by the investment amount (100,000)

In [13]:
```
for x in tickers:
    portt[f"{x} Pos"] = portt[f"{x} Allocation"] * 100000
```

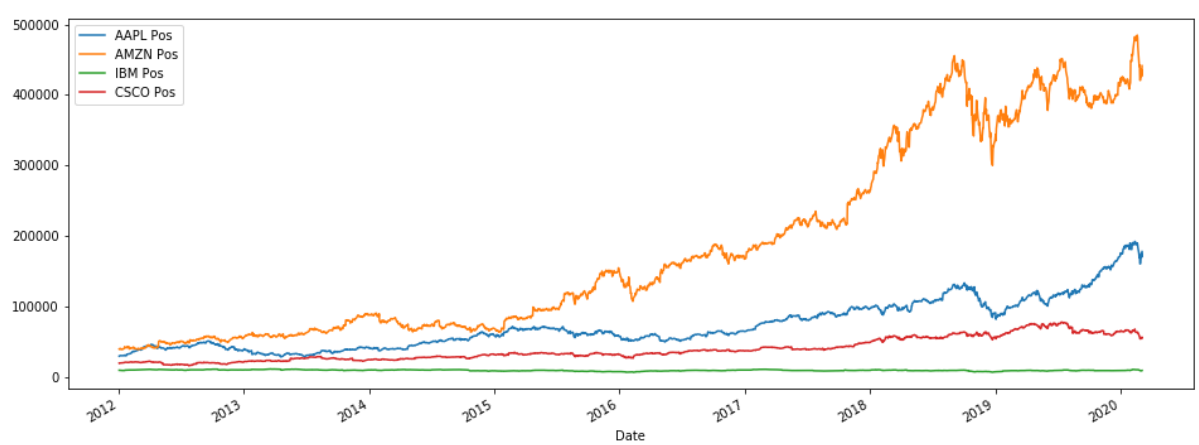
In [14]:
```
portt.head()
```

Out[14]:

Date	AAPL	AMZN	IBM	CSCO	AAPL normal return	AMZN normal return	IBM normal return	CSCO normal return	All
2012-01-03	50.994907	179.029999	139.934006	14.633397	1.000000	1.000000	1.000000	1.000000	0
2012-01-04	51.268970	177.509995	139.363144	14.916168	1.005374	0.991510	0.995920	1.019324	0
2012-01-05	51.838169	177.610001	138.702209	14.861184	1.016536	0.992068	0.991197	1.015566	0
2012-01-06	52.380054	182.610001	137.109772	14.806202	1.027162	1.019997	0.979817	1.011809	0
2012-01-09	52.296970	178.559998	136.396225	14.900459	1.025533	0.997375	0.974718	1.018250	0

In [15]:
```
portt[['AAPL Pos', 'AMZN Pos', 'IBM Pos', 'CSCO Pos']].plot(figsize = (1
6,6))
```

Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x1c19e819e8>



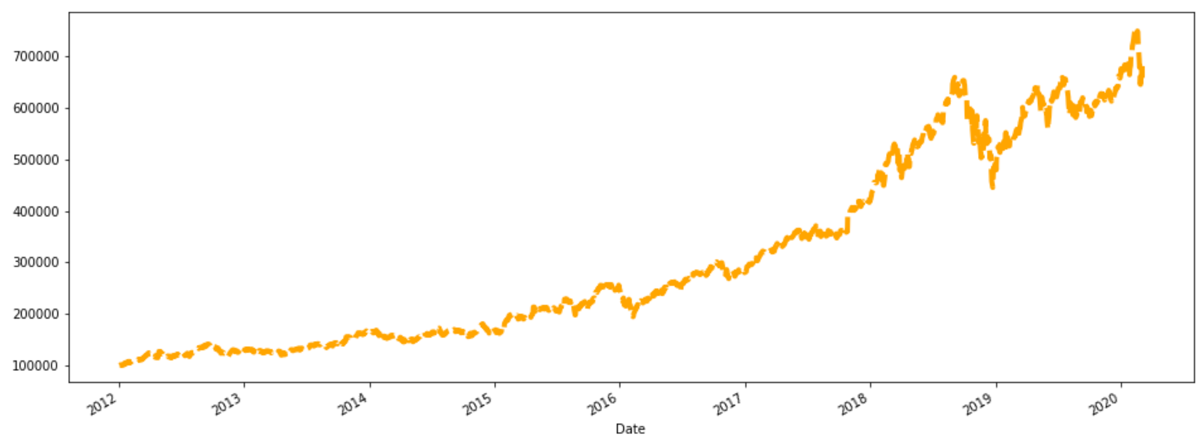
With the portfolio weighed the way we have Amazon would have given you a return of up to 500,000 in 2020 with an investment of 100,000 in the portfolio. We can see that there is some volility in the portfolio from 2018 to mid 2019. We will calculate the total Positon for the portfolio below

Lets calculate the total pos for the portfolio daily to track the movement/growth of the portfolio

In [16]:
```
portt["Total Pos"] = portt[['AAPL Pos','AMZN Pos','IBM Pos','CSCO Pos']]
.sum(axis = 1)
```

We can now track the growth of our position for a portfolio investment of 100,000 over the period. WE see below that this portfolio would have given us a return of a high of around 800,000 in 2020. Not bad for a 8 year span and a 100,000 investment. WE are looking at 8X our original investment.

In [17]:
```
portt["Total Pos"].plot(figsize = (16,6), ls = "--", lw = 4, c = "orang
e");
```



Lets calculate the daily returns and average daily returns

In [18]:
```
portt["Daily Return"] = portt["Total Pos"].pct_change()
```

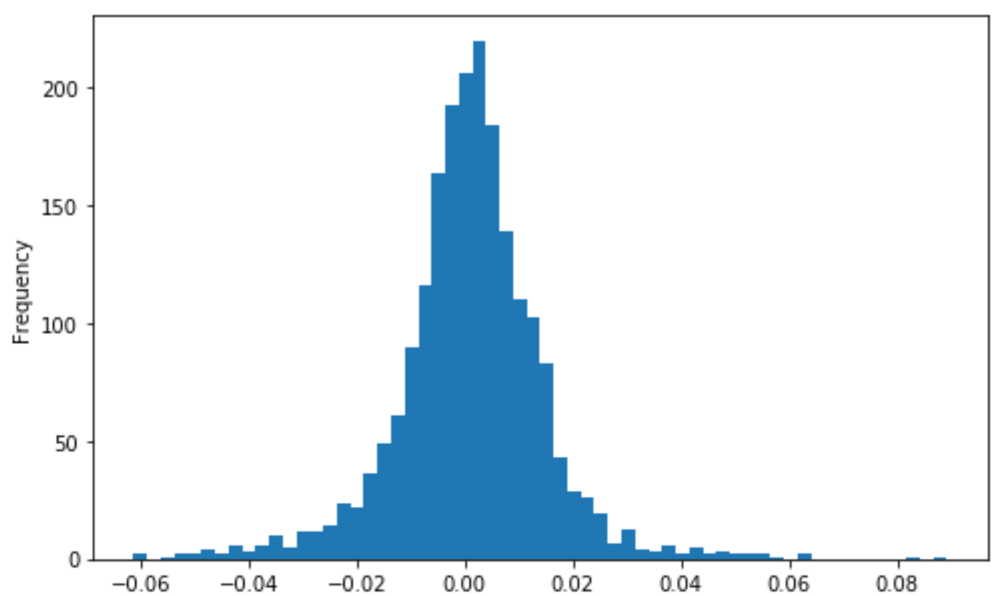
In [19]:
```
portt.head()
```

Out[19]:

Date	AAPL	AMZN	IBM	CSCO	AAPL normal return	AMZN normal return	IBM normal return	CSCO normal return	All
2012-01-03	50.994907	179.029999	139.934006	14.633397	1.000000	1.000000	1.000000	1.000000	0
2012-01-04	51.268970	177.509995	139.363144	14.916168	1.005374	0.991510	0.995920	1.019324	0
2012-01-05	51.838169	177.610001	138.702209	14.861184	1.016536	0.992068	0.991197	1.015566	0
2012-01-06	52.380054	182.610001	137.109772	14.806202	1.027162	1.019997	0.979817	1.011809	0
2012-01-09	52.296970	178.559998	136.396225	14.900459	1.025533	0.997375	0.974718	1.018250	0

In [20]:
```
portt["Daily Return"].plot(kind = "hist", bins = 60, figsize = (8,5))
```

Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x1c19e81f28>



Above we see a normal distribution in the Volitility for the portfolio. We can see that the volitility of the daily return has a low range od -.06 and a high range of .08. THis is to say that this portfolio is not as volitility nd should in end have a pretty good sharp ratio. We will calculate this below.

In [21]:
```
avg_daily_return = portt["Daily Return"].mean()
avg_daily_return
```

Out[21]: 0.0010316115174737648

In [22]:
```
port_std = portt["Daily Return"].std()
port_std
```

Out[22]: 0.013711192235850862

Let calculate the sharp ratio for the portfolio using the average daily return as well as standard deviation/Volitility of the portfolio. Annual Sharp ratio is the Risk adjusted return for the portfolio assuming risk free rate is 0

In [23]:
```
SR = (portt["Daily Return"].mean()/ portt["Daily Return"].std()) * 252**
0.5
```

In [24]:
```
SR
```

Out[24]: 1.194376453009961

Annualized sharp value is ok here meaning that the volitility of the portfolio is fair. This summary is depending in the investor risk sensitivity. Overall the question here is how do we optimize the portfolio to give us the most effieficient portfolio with the optimal Weights/Allocations. We will explore this below

In [25]:
```
import numpy as np
```

lets get the mean daily returns for each stock

In [26]:
```
por_Daily_Returns = portt[['AAPL', 'AMZN', 'IBM', 'CSCO']].pct_change(1)
```

In [27]:
```
por_Daily_Returns.mean()
```

Out[27]:
```
AAPL    0.001002
AMZN    0.001344
IBM     0.000058
CSCO    0.000611
dtype: float64
```

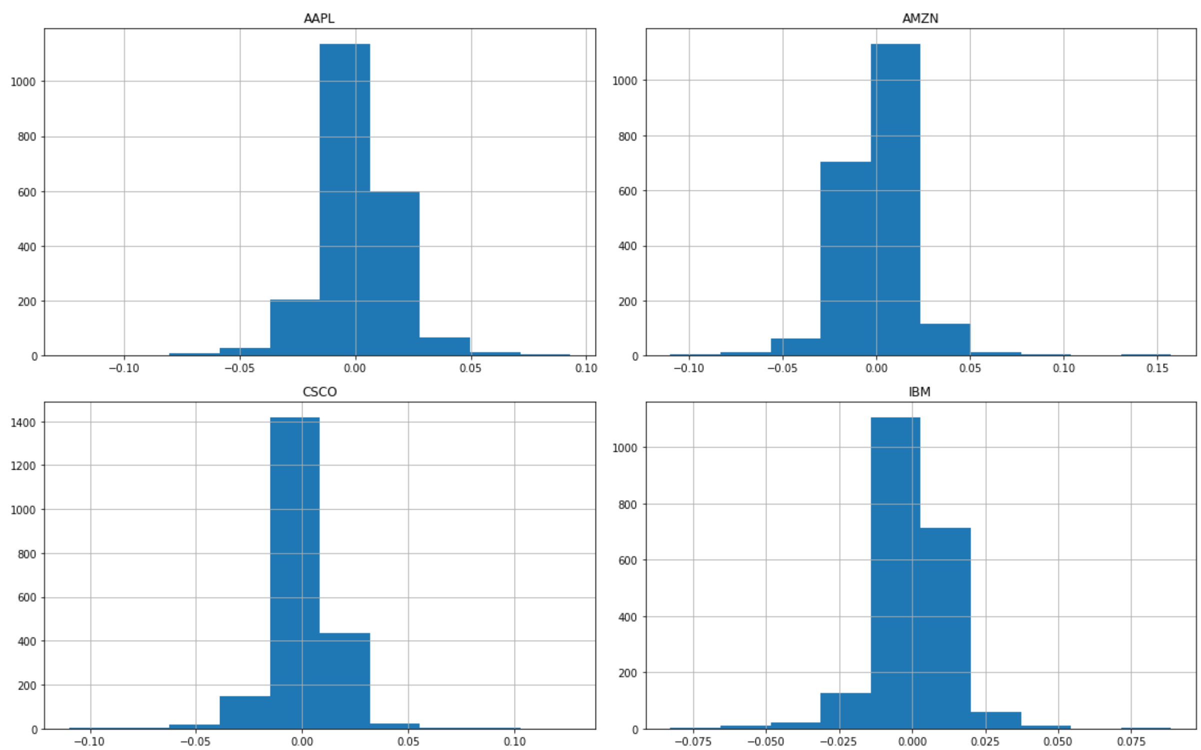
Below ew see if there is a slight corrleation between the average daily returns of IBM and CSCO but not enough to be significant. Other then that we see there is not much of a correlation between the daily returns for stocks in this portfolio. This is a good sign for diversificaiton purposes.

In [28]:
```
por_Daily_Returns.corr()
```

Out[28]:

	AAPL	AMZN	IBM	CSCO
AAPL	1.000000	0.362350	0.340776	0.402115
AMZN	0.362350	1.000000	0.311733	0.369962
IBM	0.340776	0.311733	1.000000	0.454481
CSCO	0.402115	0.369962	0.454481	1.000000

In [29]:
```
por_Daily_Returns.hist(figsize = (16,10))
plt.tight_layout()
```



We have asked the question of, how do we get the most efficient portfolio. Where the sharp ratio is the best for our liking. Again this will all depend on an investors taste for risk. We still, on the other hand, will need to find the best weights that will satisfiy the investor liking for his or her level of risk. What we will do is take 3000 portfolios that are randomally weighted and calculate the sharp ratio. Using Markowitz effieicnt frontier we will find a few portfolios that will match to the personality of an investor.

lets begin with setting up random allocations

In [30]:
```
import numpy as np
```

In [31]:
```
stocks  = portt[['AAPL', 'AMZN', 'IBM', 'CSCO']]
```

In [32]:
```
number_port = 3000
## All random weights will go here
all_weights = np.zeros((number_port, len(stocks.columns)))
## All expected returns will go here
returns_arr = np.zeros(number_port)
## All 3000 portfolios volitilities will go here
vol_arr = np.zeros(number_port)
## All Sharp ratios will go below
sharp_array = np.zeros(number_port)

## We will loop 3000 portfolios
for ind in range(number_port):

    ## Will need to calculate all weights that add up to 1
    wee = np.array(np.random.random(4))
    wee = wee/wee.sum()

    ## save weights to array
    all_weights[ind,:] = wee

    ##Calculate expected returns
    returns_arr[ind] = np.sum(por_Daily_Returns.mean() * wee *252)
    ## Calculate Portfolios Volitility
    vol_arr[ind] = np.sqrt(np.dot(wee.T, np.dot(por_Daily_Returns.cov()
* 252, wee)))
    ##Calculate the sharp ratios
    sharp_array[ind] = returns_arr[ind]/vol_arr[ind]
```

Below we will get the max sharp ratio and its locaiton in the array. This will give us the most efficient portfolio at the desired weights

In [33]:
```
sharp_array.max()
```
Out[33]:
```
1.2862911250053801
```

In [34]:
```
sharp_array.argmax()
```
Out[34]:
```
226
```

Weights below are associated with the max sharp of 1.27

In [35]:
```
all_weights[1658]
```
Out[35]:
```
array([0.25372334, 0.1884512 , 0.35347847, 0.20434699])
```

Below are the points of the highest returns for a portfolio and lowest volitility/risk.

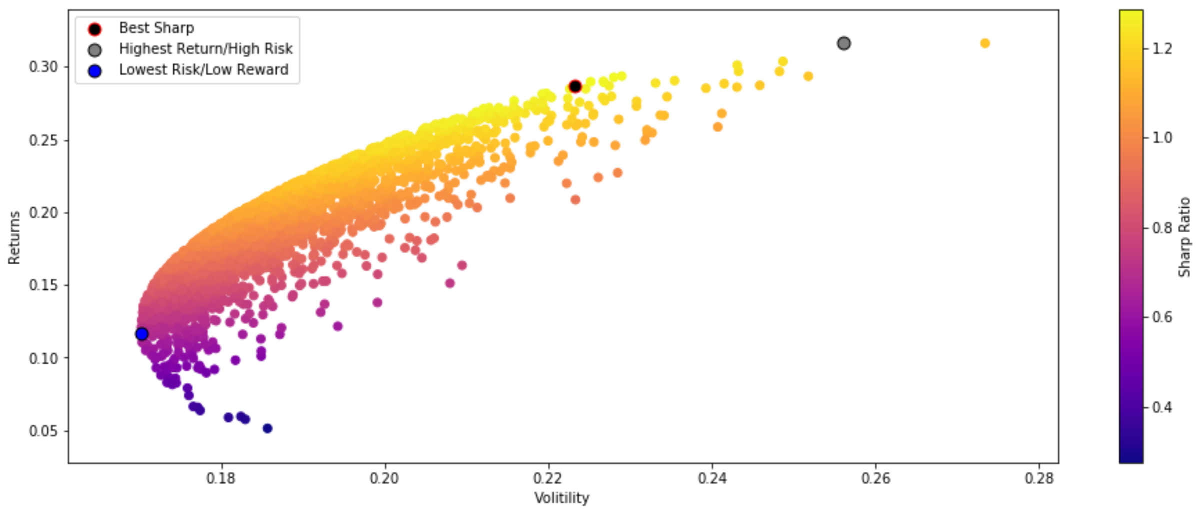
In [36]:
```
returns_arr.argmax()
```
Out[36]:
```
1545
```

In [37]:
```
vol_arr.argmin()
```
Out[37]:
```
1476
```

We will plot all portfolios as well as the point of the highest sharp ratio, lowest Risk, and the highest returns. Agian these are the managers/investors personalities we are looking to cater to.

In [44]:
```
plt.figure(figsize=(16,6))
plt.scatter(vol_arr, returns_arr, c=sharp_array, cmap="plasma")
plt.colorbar(label = "Sharp Ratio")
plt.xlabel("Volitility")
plt.ylabel("Returns")
plt.scatter(vol_arr[226], returns_arr[226], c = "black", s = 80, edgecol
or = "red", label = "Best Sharp")
plt.scatter(vol_arr[1545], returns_arr[1545], c = "grey", s = 80, edgeco
lor = "black", label = "Highest Return/High Risk")
plt.scatter(vol_arr[1476], returns_arr[1476], c = "blue", s = 80, edgeco
lor = "black", label = "Lowest Risk/Low Reward")
plt.legend()
```

Out[44]: <matplotlib.legend.Legend at 0x1c1b263be0>



In summary the Markkowitz efficient frontier tells us that there is a certain portfolio containing certain stocks that are weighted a certain way that will be the most efficient. Meaning we will have a risk adjusted return that will give us the the highest possible return for a minimal risk.

So we need to consider a few things as an investor. Do we not care about risk and will take out chances on a highly volitile portfolio but get a high return

Do we not want much risk at all and will sacrafice a larger return to avoid a higher risk.

Or are we in between. Willing to take on some risk for a moderate return.

This all depends who you are and your taste of risk is as an investor.

In [ ]: