In [194]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import quandl
import scipy.optimize as sco
from scipy.optimize import minimize
plt.style.use('fivethirtyeight')
np.random.seed(777)
%matplotlib inline
%config InlineBackend.figure_format = 'retina'
```

In [223]:

```
num_stocks = int(input("Number of stocks in the portfolio?"))

number = ['first', 'second', 'thirth', 'fourth', 'fifth', 'sixth', 'seven
th', 'eight', 'ninth', 'tenth']
stocks = []

for num in range (num_stocks):
    stocks.append(input("Name of the " + number[num] + " stock?"))
print(stocks)
```

```
Number of stocks in the portfolio?7
Name of the first stock?AAPL
Name of the second stock?AMZN
Name of the thirth stock?GOOGL
Name of the fourth stock?FB
Name of the fifth stock?TSLA
Name of the sixth stock?MSFT
Name of the seventh stock?TWTR
['AAPL', 'AMZN', 'GOOGL', 'FB', 'TSLA', 'MSFT', 'TWTR']
```

In [224]:

```
investment_duration = float(input('How many years would this portfolio in
vestment hold? (can be fraction)'))
expected_annual_returns = []
for num in range(num_stocks):
    expected_annual_returns.append(float(input("The expected return of "
+ stocks[num] + " in the investment period?"))/(investment_duration*252))

return_target = float(input("What is the target return in the investment
 period?"))/investment_duration
print(expected_annual_returns)
```

```
How many years would this portfolio investment hold? (can be
fraction)1
The expected return of AAPL in the investment period?0.33
The expected return of AMZN in the investment period?0.25
The expected return of GOOGL in the investment period?0.17
The expected return of FB in the investment period?0.14
The expected return of TSLA in the investment period?0.24
The expected return of MSFT in the investment period?0.27
The expected return of TWTR in the investment period?0.3
What is the target return in the investment period?0.27
[0.0013095238095238097, 0.000992063492063492, 0.0006746031746
031747, 0.0005555555555555556, 0.0009523809523809524, 0.00107
14285714285715, 0.0011904761904761904]
```

In [229]:

```
quandl.ApiConfig.api_key = 'PrW6L55BexiSBEqWU25u'
data = quandl.get_table('WIKI/PRICES', ticker = stocks,
                        qopts = { 'columns': ['date', 'ticker', 'adj_clos
e'] },
                        date = { 'gte': '2016-1-1', 'lte': '2017-12-31'
}, paginate=True)
data.head()
```

Out[229]:

	date	ticker	adj_close
None			
0	2017-12-29	TWTR	24.01
1	2017-12-28	TWTR	24.31
2	2017-12-27	TWTR	24.23
3	2017-12-26	TWTR	24.26
4	2017-12-22	TWTR	24.46

In [230]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3512 entries, 0 to 3511
Data columns (total 3 columns):
date         3512 non-null datetime64[ns]
ticker       3512 non-null object
adj_close    3512 non-null float64
dtypes: datetime64[ns](1), float64(1), object(1)
memory usage: 82.4+ KB
```

In [231]:

```
df = data.set_index('date')
table = df.pivot(columns='ticker')
# By specifying col[1] in below list comprehension
# You can select the stock names under multi-level column
table.columns = [col[1] for col in table.columns]
table.head()
```

Out[231]:

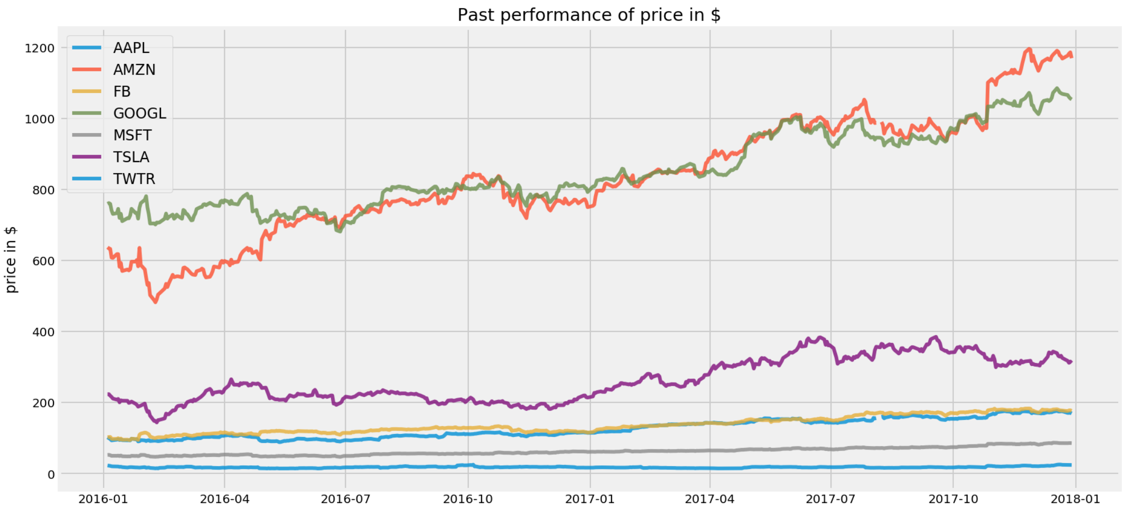
	AAPL	AMZN	FB	GOOGL	MSFT	TSLA	TWTR
date							
2016-01-04	101.783763	636.99	102.22	759.44	52.181598	223.41	22.56
2016-01-05	99.233131	633.79	102.73	761.53	52.419653	223.43	21.92
2016-01-06	97.291172	632.65	102.97	759.33	51.467434	219.04	21.39
2016-01-07	93.185040	607.94	97.92	741.00	49.677262	215.65	20.26
2016-01-08	93.677776	607.05	97.33	730.91	49.829617	211.00	19.98

In [290]:

```
plt.figure(figsize=(14, num_stocks))
for c in table.columns.values:
    plt.plot(table.index, table[c], lw=3, alpha=0.8,label=c)
plt.legend(loc='upper left', fontsize=12)
plt.ylabel('price in $')
plt.title('Past performance of price in $')
```

Out[290]:

```
Text(0.5, 1.0, 'Past performance of price in $')
```

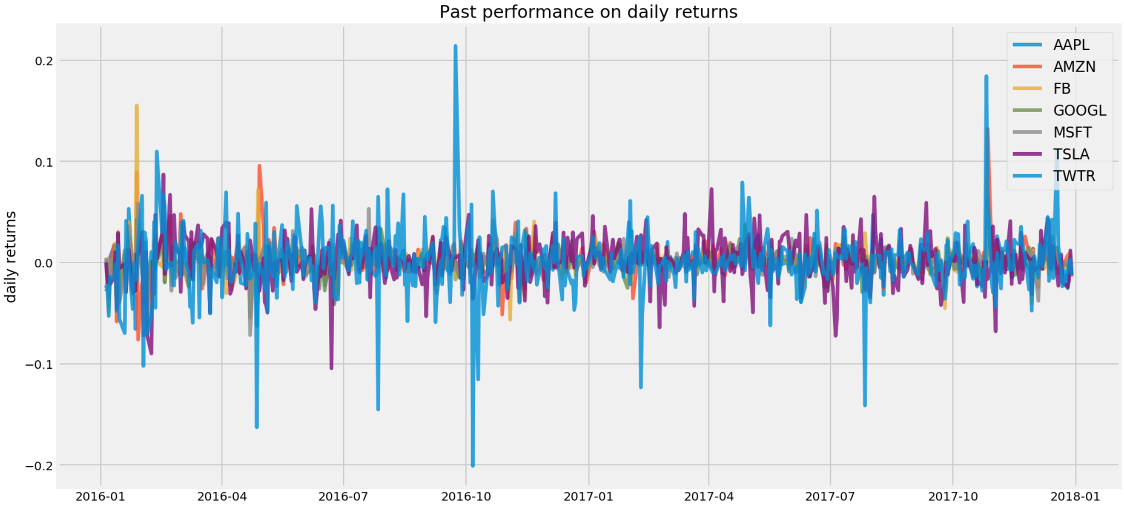


In [289]:

```
returns = table.pct_change()
plt.figure(figsize=(14, num_stocks))
for c in returns.columns.values:
    plt.plot(returns.index, returns[c], lw=3, alpha=0.8,label=c)
plt.legend(loc='upper right', fontsize=12)
plt.ylabel('daily returns')
plt.title('Past performance on daily returns')
```

Out[289]:

```
Text(0.5, 1.0, 'Past performance on daily returns')
```



In [234]:

```
def portfolio_annualised_performance(weights, mean_returns, cov_matrix):
    returns = np.sum(mean_returns*weights ) *252
    std = np.sqrt(np.dot(weights.T, np.dot(cov_matrix, weights))) * np.sq
rt(252)
    return std, returns

def random_portfolios(num_portfolios, mean_returns, cov_matrix, risk_free
_rate):
    results = np.zeros((3,num_portfolios))
    weights_record = []
    for i in range(num_portfolios):
        weights = np.random.random(num_stocks)
        weights /= np.sum(weights)
        weights_record.append(weights)
        portfolio_std_dev, portfolio_return = portfolio_annualised_perfor
mance(weights, mean_returns, cov_matrix)
        results[0,i] = portfolio_std_dev
        results[1,i] = portfolio_return
        results[2,i] = (portfolio_return - risk_free_rate) / portfolio_st
d_dev
    return results, weights_record
```

In [297]:

```
returns = table.pct_change()
mean_returns = returns.mean()
expected_returns = pd.Series(expected_annual_returns, index = stocks)
cov_matrix = returns.cov()
num_portfolios = 100000
risk_free_rate = 0.0178

print("mean_return(daily) in the past 2 years:")
print(mean_returns)
print('\n')
print("expected_returns(daily) in the investment period:")
print(expected_returns)
type(mean_returns)
```

```
mean_return(daily) in the past 2 years:
AAPL     0.001101
AMZN     0.001340
FB       0.001197
GOOGL    0.000716
MSFT     0.001060
TSLA     0.000933
TWTR     0.000662
dtype: float64


expected_returns(daily) in the investment period:
AAPL     0.001310
AMZN     0.000992
GOOGL    0.000675
FB       0.000556
TSLA     0.000952
MSFT     0.001071
TWTR     0.001190
dtype: float64
```

Out[297]:

```
pandas.core.series.Series
```

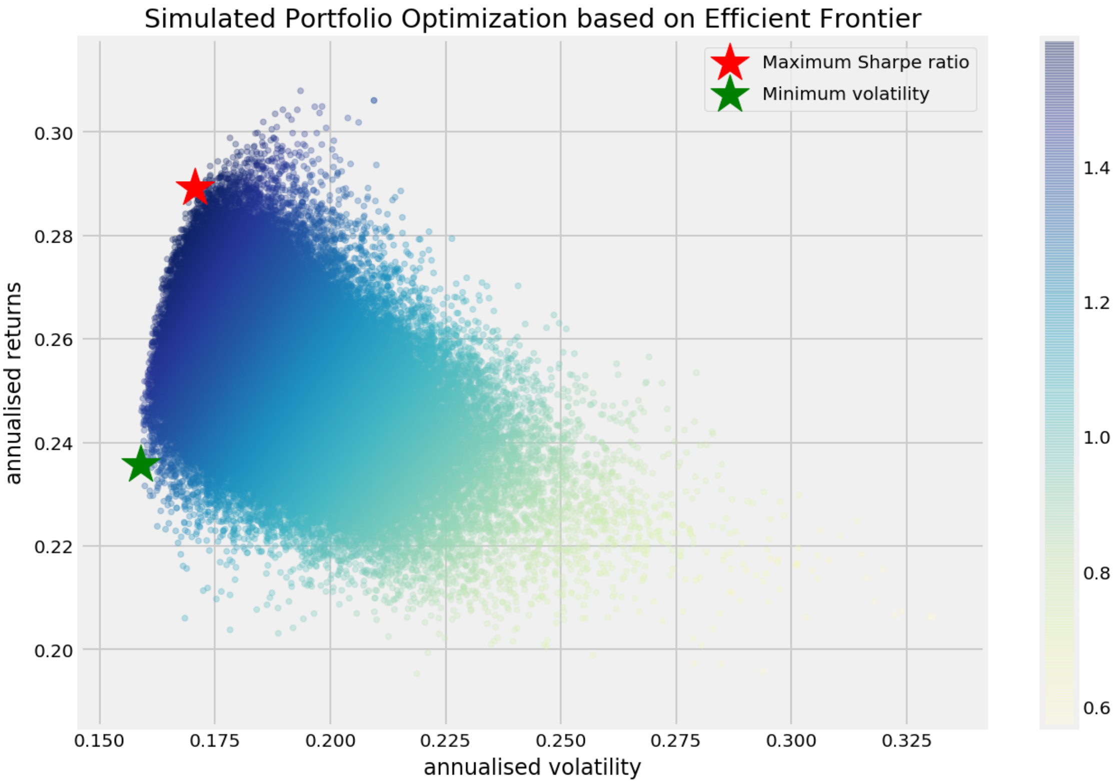
In [284]:

```
def display_simulated_ef_with_random(mean_returns, cov_matrix, num_portfolios, risk_free_rate, return_target=0):
    results, weights = random_portfolios(num_portfolios,mean_returns, cov_matrix, risk_free_rate)

    max_sharpe_idx = np.argmax(results[2])
    sdp, rp = results[0,max_sharpe_idx], results[1,max_sharpe_idx]
    max_sharpe_allocation = pd.DataFrame(weights[max_sharpe_idx],index=table.columns,columns=['allocation'])
    max_sharpe_allocation.allocation = [round(i*100,2)for i in max_sharpe_allocation.allocation]
    max_sharpe_allocation = max_sharpe_allocation.T

    min_vol_idx = np.argmin(results[0])
    sdp_min, rp_min = results[0,min_vol_idx], results[1,min_vol_idx]
    min_vol_allocation = pd.DataFrame(weights[min_vol_idx],index=table.columns,columns=['allocation'])
    min_vol_allocation.allocation = [round(i*100,2)for i in min_vol_allocation.allocation]
    min_vol_allocation = min_vol_allocation.T

    if return_target!=0:
        target_min_vol = efficient_return(expected_returns, cov_matrix, return_target)
        target_sdp_min, target_rp_min = portfolio_annualised_performance(target_min_vol['x'], mean_returns, cov_matrix)
        target_min_vol_allocation = pd.DataFrame(target_min_vol.x,index=table.columns,columns=['allocation'])
        target_min_vol_allocation.allocation = [round(i*100,2)for i in target_min_vol_allocation.allocation]
        target_min_vol_allocation = target_min_vol_allocation.T

    print ("-"*80)
    print ("Maximum Sharpe Ratio Portfolio Allocation\n")
    print ("Annualised Return:", round(rp,2))
    print ("Annualised Volatility:", round(sdp,2))
    print ("\n")
    print (max_sharpe_allocation)
    print ("-"*80)
    print ("Minimum Volatility Portfolio Allocation\n")
    print ("Annualised Return:", round(rp_min,2))
    print ("Annualised Volatility:", round(sdp_min,2))
    print ("\n")
    print (min_vol_allocation)

    if return_target!=0:
        print ("-"*80)
        print ("Minimum Volatility of The Target Return Portfolio Allocation\n")
        print ("Annualised Return:", round(target_rp_min,2))
        print ("Annualised Volatility:", round(target_sdp_min,2))
        print ("\n")
        print (target_min_vol_allocation)
    plt.figure(figsize=(10, num_stocks))
    plt.scatter(results[0,:],results[1,:],c=results[2,:],cmap='YlGnBu', marker='o', s=10, alpha=0.3)
    plt.colorbar()
    plt.scatter(sdp,rp,marker='*',color='r',s=500, label='Maximum Sharpe ratio')
    plt.scatter(sdp_min,rp_min,marker='*',color='g',s=500, label='Minimum volatility')
    if return_target!=0:
        plt.scatter(efficient_return(expected_returns, cov_matrix, return_target).fun ,return_target,marker='*',color='b',s=500, label='Minimum volatility of the target return')
    plt.title('Simulated Portfolio Optimization based on Efficient Frontier')
    plt.xlabel('annualised volatility')
    plt.ylabel('annualised returns')
    plt.legend(labelspacing=0.8)
```

In [312]:

```
display_simulated_ef_with_random(mean_returns, cov_matrix, num_portfolios
, risk_free_rate)
print ("-"*80)
print("\n\n***It is a optimization base on the past performance in return
***")
```

```
---------------------------------------------------------------
-------------------
Maximum Sharpe Ratio Portfolio Allocation

Annualised Return: 0.29
Annualised Volatility: 0.17


             AAPL   AMZN     FB  GOOGL   MSFT  TSLA  TWTR
allocation  29.07  21.31  19.08   1.19  27.58  0.57   1.2
---------------------------------------------------------------
-------------------
Minimum Volatility Portfolio Allocation

Annualised Return: 0.24
Annualised Volatility: 0.16


            AAPL  AMZN    FB  GOOGL   MSFT  TSLA  TWTR
allocation  29.2  1.89  0.31  35.51  24.15  5.77  3.18
---------------------------------------------------------------
-------------------


***It is a optimization base on the past performance in retur
n***
```



In [254]:

```
def con1(x):
    return p.sum(x) - 1
```

In [255]:

```
constraints = {'type': 'eq', 'fun': con1}
```

In [256]:

```
def neg_sharpe_ratio(weights, mean_returns, cov_matrix, risk_free_rate):
    p_var, p_ret = portfolio_annualised_performance(weights, mean_returns
, cov_matrix)
    return -(p_ret - risk_free_rate) / p_var

def max_sharpe_ratio(mean_returns, cov_matrix, risk_free_rate):
    num_assets = len(mean_returns)
    args = (mean_returns, cov_matrix, risk_free_rate)
    constraints = ({'type': 'eq', 'fun': lambda x: np.sum(x) - 1})
    bound = (0.0,1.0)
    bounds = tuple(bound for asset in range(num_assets))
    result = sco.minimize(neg_sharpe_ratio, num_assets*[1./num_assets,],
args=args,
                        method='SLSQP', bounds=bounds, constraints=constr
aints)
    return result
```

In [257]:

```
def portfolio_volatility(weights, mean_returns, cov_matrix):
    return portfolio_annualised_performance(weights, mean_returns, cov_ma
trix)[0]

def min_variance(mean_returns, cov_matrix):
    num_assets = len(mean_returns)
    args = (mean_returns, cov_matrix)
    constraints = ({'type': 'eq', 'fun': lambda x: np.sum(x) - 1})
    bound = (0.0,1.0)
    bounds = tuple(bound for asset in range(num_assets))

    result = sco.minimize(portfolio_volatility, num_assets*[1./num_assets
,], args=args,
                        method='SLSQP', bounds=bounds, constraints=constr
aints)

    return result
```

In [258]:

```
def efficient_return(mean_returns, cov_matrix, target):
    num_assets = len(mean_returns)
    args = (mean_returns, cov_matrix)

    def portfolio_return(weights):
        return portfolio_annualised_performance(weights, mean_returns, co
v_matrix)[1]

    constraints = ({'type': 'eq', 'fun': lambda x: portfolio_return(x) -
target},
                   {'type': 'eq', 'fun': lambda x: np.sum(x) - 1})
    bounds = tuple((0,1) for asset in range(num_assets))
    result = sco.minimize(portfolio_volatility, num_assets*[1./num_assets
,], args=args, method='SLSQP', bounds=bounds, constraints=constraints)
    return result


def efficient_frontier(mean_returns, cov_matrix, returns_range):
    efficients = []
    for ret in returns_range:
        efficients.append(efficient_return(mean_returns, cov_matrix, ret
))
    return efficients
```

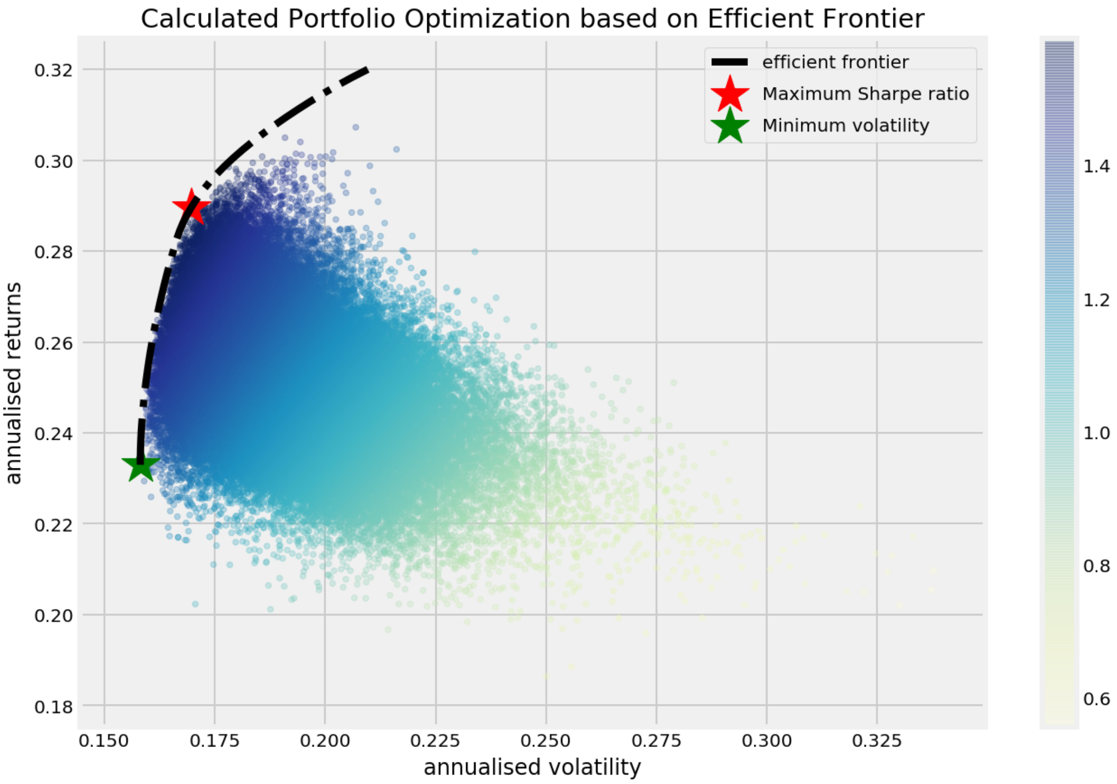
In [282]:

```
def display_calculated_ef_with_random(mean_returns, cov_matrix, num_portfolios, risk_free_rate, return_target=0):
    results, _ = random_portfolios(num_portfolios,mean_returns, cov_matrix, risk_free_rate)

    max_sharpe = max_sharpe_ratio(mean_returns, cov_matrix, risk_free_rate)
    sdp, rp = portfolio_annualised_performance(max_sharpe['x'], mean_returns, cov_matrix)
    max_sharpe_allocation = pd.DataFrame(max_sharpe.x,index=table.columns,columns=['allocation'])
    max_sharpe_allocation.allocation = [round(i*100,2)for i in max_sharpe_allocation.allocation]
    max_sharpe_allocation = max_sharpe_allocation.T

    min_vol = min_variance(mean_returns, cov_matrix)
    sdp_min, rp_min = portfolio_annualised_performance(min_vol['x'], mean_returns, cov_matrix)
    min_vol_allocation = pd.DataFrame(min_vol.x,index=table.columns,columns=['allocation'])
    min_vol_allocation.allocation = [round(i*100,2)for i in min_vol_allocation.allocation]
    min_vol_allocation = min_vol_allocation.T

    if return_target!=0:
        target_min_vol = efficient_return(expected_returns, cov_matrix, return_target)
        target_sdp_min, target_rp_min = portfolio_annualised_performance(target_min_vol['x'], mean_returns, cov_matrix)
        target_min_vol_allocation = pd.DataFrame(target_min_vol.x,index=table.columns,columns=['allocation'])
        target_min_vol_allocation.allocation = [round(i*100,2)for i in target_min_vol_allocation.allocation]
        target_min_vol_allocation = target_min_vol_allocation.T

    print ("-"*80)
    print ("Maximum Sharpe Ratio Portfolio Allocation\n")
    print ("Annualised Return:", round(rp,2))
    print ("Annualised Volatility:", round(sdp,2))
    print ("\n")
    print (max_sharpe_allocation)
    print ("-"*80)
    print ("Minimum Volatility Portfolio Allocation\n")
    print ("Annualised Return:", round(rp_min,2))
    print ("Annualised Volatility:", round(sdp_min,2))
    print ("\n")
    print (min_vol_allocation)

    if return_target!=0:
        print ("-"*80)
        print ("Minimum Volatility of The Target Return Portfolio Allocation\n")
        print ("Annualised Return:", round(target_rp_min,2))
        print ("Annualised Volatility:", round(target_sdp_min,2))
        print ("\n")
        print (target_min_vol_allocation)

    plt.figure(figsize=(10, num_stocks))
    plt.scatter(results[0,:],results[1,:],c=results[2,:],cmap='YlGnBu', m
arker='o', s=10, alpha=0.3)
    plt.colorbar()
    plt.scatter(sdp,rp,marker='*',color='r',s=500, label='Maximum Sharpe
 ratio')
    plt.scatter(sdp_min,rp_min,marker='*',color='g',s=500, label='Minimum
volatility')
    if return_target!=0:
        plt.scatter(efficient_return(expected_returns, cov_matrix, return
_target).fun, return_target, marker='*',color='b',s=500, label='Minimum v
olatility of the target return')
    target = np.linspace(rp_min, 0.32, 50)
    efficient_portfolios = efficient_frontier(mean_returns, cov_matrix, t
arget)
    plt.plot([p['fun'] for p in efficient_portfolios], target, linestyle=
'-.', color='black', label='efficient frontier')
    plt.title('Calculated Portfolio Optimization based on Efficient Front
ier')
    plt.xlabel('annualised volatility')
    plt.ylabel('annualised returns')
    plt.legend(labelspacing=0.8)
```

In [313]:

```
display_calculated_ef_with_random(mean_returns, cov_matrix, num_portfolios, risk_free_rate)
print ("-"*80)
print("\n\n***It is a optimization base on the past performance in return***")
```

```
--------------------------------------------------------------
-------------------
Maximum Sharpe Ratio Portfolio Allocation

Annualised Return: 0.29
Annualised Volatility: 0.17


             AAPL   AMZN     FB  GOOGL   MSFT  TSLA  TWTR
allocation  32.42  19.51  17.01    0.0  29.82  1.24   0.0
--------------------------------------------------------------
-------------------
Minimum Volatility Portfolio Allocation

Annualised Return: 0.23
Annualised Volatility: 0.16


             AAPL  AMZN    FB  GOOGL   MSFT  TSLA  TWTR
allocation  26.23   0.0  3.24  40.11  23.93  4.87  1.62
--------------------------------------------------------------
-------------------


***It is a optimization base on the past performance in retur
n***
```



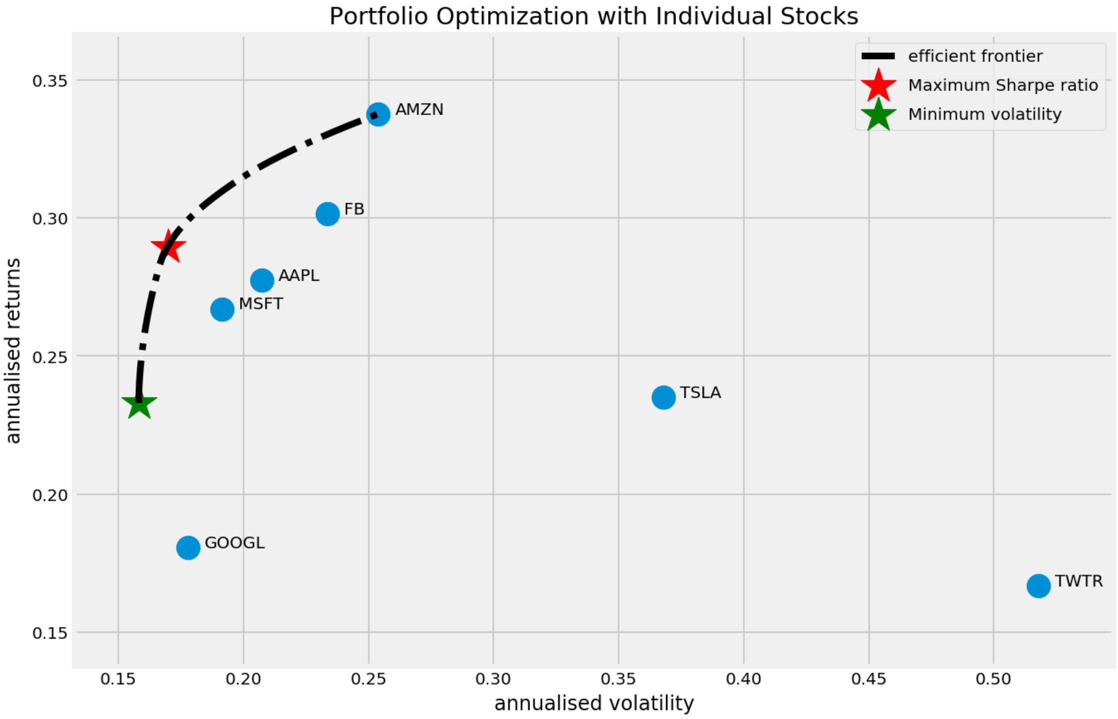
In [279]:

```
def display_ef_with_selected(mean_returns, cov_matrix, risk_free_rate, return_target=0):
    max_sharpe = max_sharpe_ratio(mean_returns, cov_matrix, risk_free_rate)
    sdp, rp = portfolio_annualised_performance(max_sharpe['x'], mean_returns, cov_matrix)
    max_sharpe_allocation = pd.DataFrame(max_sharpe.x,index=table.columns,columns=['allocation'])
    max_sharpe_allocation.allocation = [round(i*100,2)for i in max_sharpe_allocation.allocation]
    max_sharpe_allocation = max_sharpe_allocation.T

    min_vol = min_variance(mean_returns, cov_matrix)
    sdp_min, rp_min = portfolio_annualised_performance(min_vol['x'], mean_returns, cov_matrix)
    min_vol_allocation = pd.DataFrame(min_vol.x,index=table.columns,columns=['allocation'])
    min_vol_allocation.allocation = [round(i*100,2)for i in min_vol_allocation.allocation]
    min_vol_allocation = min_vol_allocation.T

    if return_target!=0:
        target_min_vol = efficient_return(expected_returns, cov_matrix, return_target)
        target_sdp_min, target_rp_min = portfolio_annualised_performance(target_min_vol['x'], mean_returns, cov_matrix)
        target_min_vol_allocation = pd.DataFrame(target_min_vol.x,index=table.columns,columns=['allocation'])
        target_min_vol_allocation.allocation = [round(i*100,2)for i in target_min_vol_allocation.allocation]
        target_min_vol_allocation = target_min_vol_allocation.T


    an_vol = np.std(returns) * np.sqrt(252)
    an_rt = mean_returns * 252

    print ("-"*80)
    print ("Maximum Sharpe Ratio Portfolio Allocation\n")
    print ("Annualised Return:", round(rp,2))
    print ("Annualised Volatility:", round(sdp,2))
    print ("\n")
    print (max_sharpe_allocation)
    print ("-"*80)
    print ("Minimum Volatility Portfolio Allocation\n")
    print ("Annualised Return:", round(rp_min,2))
    print ("Annualised Volatility:", round(sdp_min,2))
    print ("\n")
    print (min_vol_allocation)

    if return_target!=0:
        print ("-"*80)
        print ("Minimum Volatility of The Target Return Portfolio Allocation\n")
        print ("Annualised Return:", round(target_rp_min,2))
        print ("Annualised Volatility:", round(target_sdp_min,2))
        print ("\n")
        print (target_min_vol_allocation)

    print ("-"*80)
    print ("Individual Stock Returns and Volatility\n")
    for i, txt in enumerate(table.columns):
        print (txt,":","annuaised return",round(an_rt[i],2),", annualised volatility:",round(an_vol[i],2))
    print ("-"*80)

    fig, ax = plt.subplots(figsize=(10, num_stocks))
    ax.scatter(an_vol,an_rt,marker='o',s=200)

    for i, txt in enumerate(table.columns):
        ax.annotate(txt, (an_vol[i],an_rt[i]), xytext=(10,0), textcoords='offset points')
    ax.scatter(sdp,rp,marker='*',color='r',s=500, label='Maximum Sharpe ratio')
    ax.scatter(sdp_min,rp_min,marker='*',color='g',s=500, label='Minimum volatility')
    if return_target!=0:
        plt.scatter(efficient_return(expected_returns, cov_matrix, return_target).fun, return_target, marker='*',color='b',s=500, label='Minimum volatility of the target return')
    target = np.linspace(rp_min, 0.34, 50)
    efficient_portfolios = efficient_frontier(mean_returns, cov_matrix, target)
    ax.plot([p['fun'] for p in efficient_portfolios], target, linestyle='-.', color='black', label='efficient frontier')
    ax.set_title('Portfolio Optimization with Individual Stocks')
    ax.set_xlabel('annualised volatility')
    ax.set_ylabel('annualised returns')
    ax.legend(labelspacing=0.8)
```

In [314]:

```
display_ef_with_selected(mean_returns, cov_matrix, risk_free_rate)
print ("-"*80)
print("\n\n***It is a optimization base on the past performance in return ***")
```

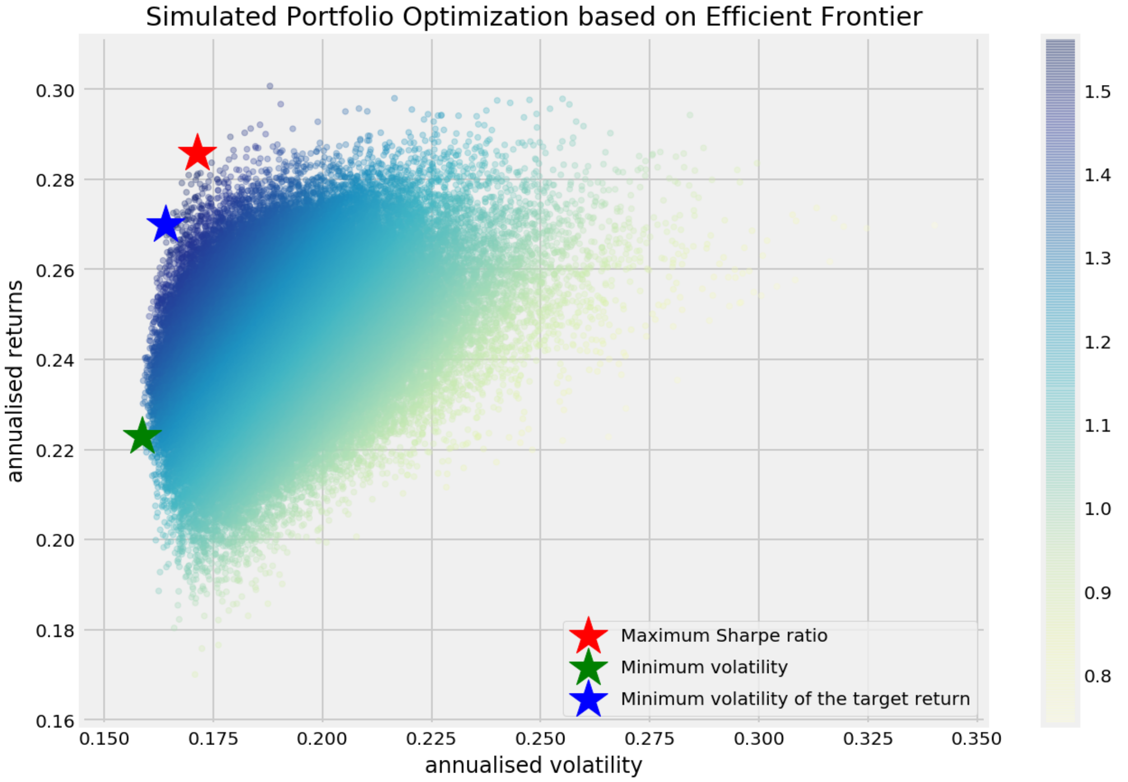
```
-------------------------------------------------------------
-------------------
Maximum Sharpe Ratio Portfolio Allocation

Annualised Return: 0.29
Annualised Volatility: 0.17


             AAPL   AMZN     FB  GOOGL   MSFT  TSLA  TWTR
allocation  32.42  19.51  17.01    0.0  29.82  1.24   0.0
-------------------------------------------------------------
-------------------
Minimum Volatility Portfolio Allocation

Annualised Return: 0.23
Annualised Volatility: 0.16


             AAPL  AMZN    FB  GOOGL   MSFT  TSLA  TWTR
allocation  26.23   0.0  3.24  40.11  23.93  4.87  1.62
-------------------------------------------------------------
-------------------
Individual Stock Returns and Volatility

AAPL : annuaised return 0.28 , annualised volatility: 0.21
AMZN : annuaised return 0.34 , annualised volatility: 0.25
FB : annuaised return 0.3 , annualised volatility: 0.23
GOOGL : annuaised return 0.18 , annualised volatility: 0.18
MSFT : annuaised return 0.27 , annualised volatility: 0.19
TSLA : annuaised return 0.24 , annualised volatility: 0.37
TWTR : annuaised return 0.17 , annualised volatility: 0.52
-------------------------------------------------------------
-------------------
-------------------------------------------------------------
-------------------


***It is a optimization base on the past performance in retur
n***
```


Portfolio Optimization with Individual Stocks
efficient frontier
Maximum Sharpe ratio
Minimum volatility
AMZN
FB
AAPL
MSFT
TSLA
GOOGL
TWTR
annualised returns
0.35
0.30
0.25
0.20
0.15
annualised volatility
0.15
0.20
0.25
0.30
0.35
0.40
0.45
0.50

In [286]:

```
display_simulated_ef_with_random(expected_returns, cov_matrix, num_portfo
lios
, risk_free_rate, return_target)
```

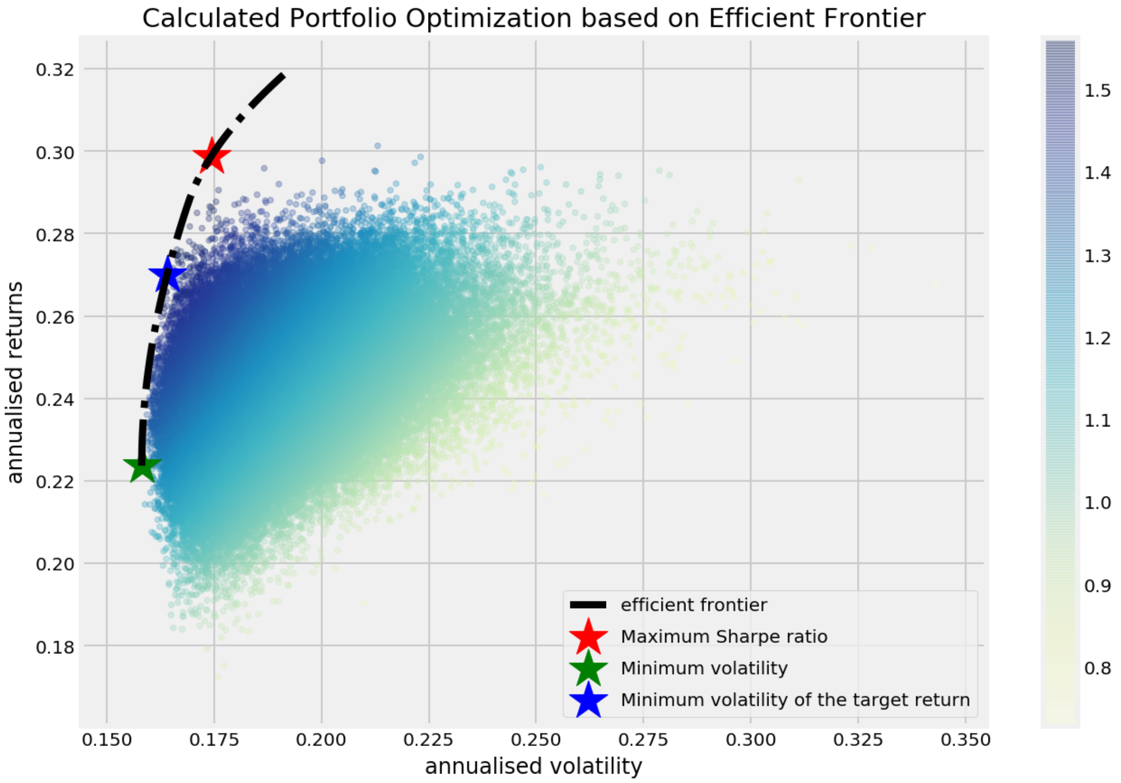
```
-------------------------------------------------------------
-------------------
Maximum Sharpe Ratio Portfolio Allocation

Annualised Return: 0.29
Annualised Volatility: 0.17


             AAPL  AMZN    FB  GOOGL   MSFT  TSLA  TWTR
allocation  47.51  0.92  2.41   2.43  31.26  7.13  8.34
-------------------------------------------------------------
-------------------
Minimum Volatility Portfolio Allocation

Annualised Return: 0.22
Annualised Volatility: 0.16


             AAPL  AMZN    FB  GOOGL  MSFT  TSLA  TWTR
allocation  22.85  1.88  3.39  37.33  28.8  5.26  0.48
-------------------------------------------------------------
-------------------
Minimum Volatility of The Target Return Portfolio Allocation

Annualised Return: 0.27
Annualised Volatility: 0.16


             AAPL  AMZN    FB  GOOGL   MSFT  TSLA  TWTR
allocation  42.67  3.99  0.24  11.77  33.13  5.96  2.23
```



In [287]:

```
display_calculated_ef_with_random(expected_returns, cov_matrix, num_portfolios, risk_free_rate, return_target)
```

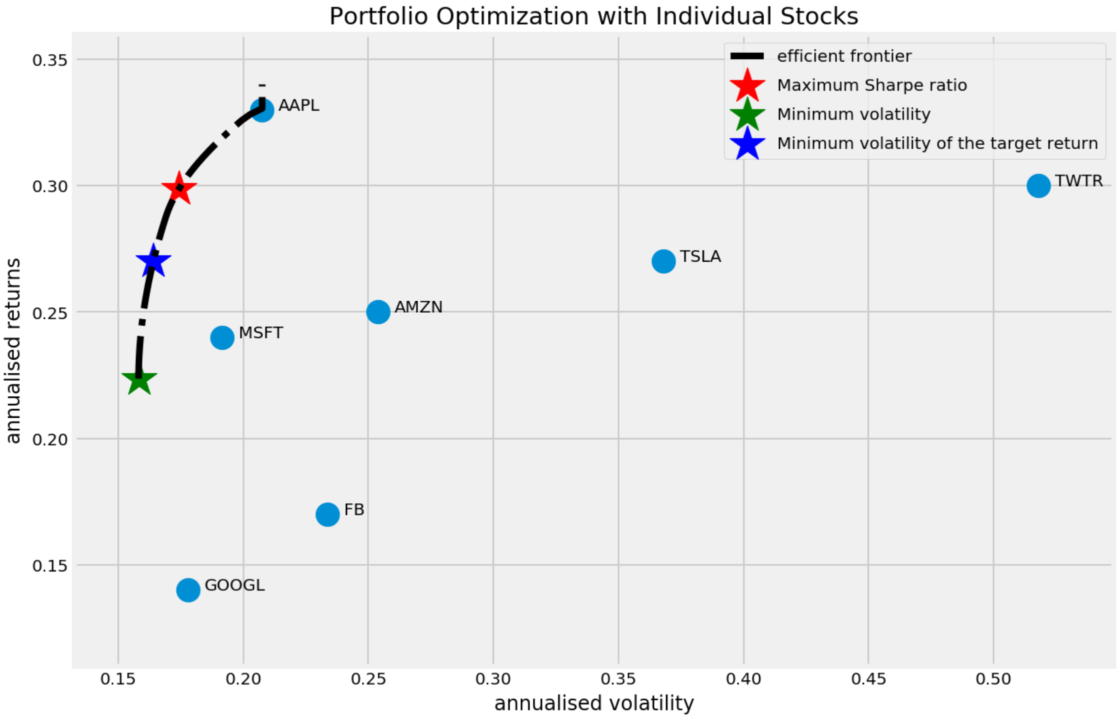
```
-------------------------------------------------------------
-------------------
Maximum Sharpe Ratio Portfolio Allocation

Annualised Return: 0.3
Annualised Volatility: 0.17


            AAPL  AMZN   FB  GOOGL   MSFT  TSLA  TWTR
allocation  60.7  4.02  0.0    0.0  26.19  5.76  3.34
-------------------------------------------------------------
-------------------
Minimum Volatility Portfolio Allocation

Annualised Return: 0.22
Annualised Volatility: 0.16


             AAPL  AMZN    FB  GOOGL   MSFT  TSLA  TWTR
allocation  26.23   0.0  3.24  40.11  23.93  4.87  1.62
-------------------------------------------------------------
-------------------
Minimum Volatility of The Target Return Portfolio Allocation

Annualised Return: 0.27
Annualised Volatility: 0.16


             AAPL  AMZN    FB  GOOGL   MSFT  TSLA  TWTR
allocation  42.67  3.99  0.24  11.77  33.13  5.96  2.23
```


Calculated Portfolio Optimization based on Efficient Frontier
annualised returns
0.32
0.30
0.28
0.26
0.24
0.22
0.20
0.18
annualised volatility
0.150
0.175
0.200
0.225
0.250
0.275
0.300
0.325
0.350
efficient frontier
Maximum Sharpe ratio
Minimum volatility
Minimum volatility of the target return
1.5
1.4
1.3
1.2
1.1
1.0
0.9
0.8

In [288]:

```
display_ef_with_selected(expected_returns, cov_matrix, risk_free_rate, re
turn_target)
```

```
-------------------------------------------------------------
-------------------
Maximum Sharpe Ratio Portfolio Allocation

Annualised Return: 0.3
Annualised Volatility: 0.17


            AAPL  AMZN   FB  GOOGL   MSFT  TSLA  TWTR
allocation  60.7  4.02  0.0    0.0  26.19  5.76  3.34
-------------------------------------------------------------
-------------------
Minimum Volatility Portfolio Allocation

Annualised Return: 0.22
Annualised Volatility: 0.16


             AAPL  AMZN    FB  GOOGL   MSFT  TSLA  TWTR
allocation  26.23   0.0  3.24  40.11  23.93  4.87  1.62
-------------------------------------------------------------
-------------------
Minimum Volatility of The Target Return Portfolio Allocation

Annualised Return: 0.27
Annualised Volatility: 0.16


             AAPL  AMZN    FB  GOOGL   MSFT  TSLA  TWTR
allocation  42.67  3.99  0.24  11.77  33.13  5.96  2.23
-------------------------------------------------------------
-------------------
Individual Stock Returns and Volatility

AAPL : annuaised return 0.33 , annualised volatility: 0.21
AMZN : annuaised return 0.25 , annualised volatility: 0.25
FB : annuaised return 0.17 , annualised volatility: 0.23
GOOGL : annuaised return 0.14 , annualised volatility: 0.18
MSFT : annuaised return 0.24 , annualised volatility: 0.19
TSLA : annuaised return 0.27 , annualised volatility: 0.37
TWTR : annuaised return 0.3 , annualised volatility: 0.52
-------------------------------------------------------------
-------------------
```


Portfolio Optimization with Individual Stocks
efficient frontier
Maximum Sharpe ratio
Minimum volatility
Minimum volatility of the target return
AAPL
TWTR
TSLA
AMZN
MSFT
FB
GOOGL
annualised returns
annualised volatility
0.35
0.30
0.25
0.20
0.15
0.15
0.20
0.25
0.30
0.35
0.40
0.45
0.50