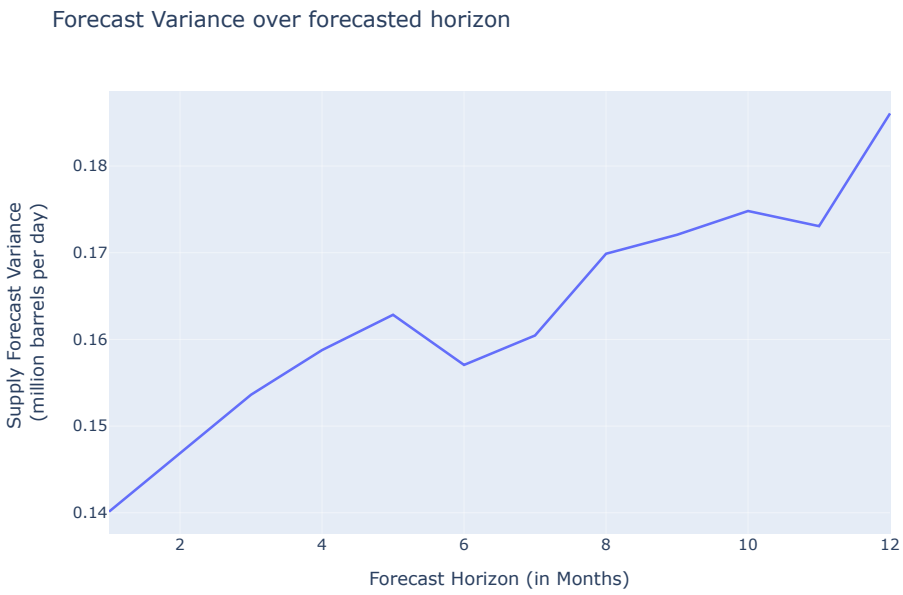
In [1]:
```
import pandas, os, glob, datetime
import plotly.express as px
import plotly.graph_objects as go
BASE_DIR='./processed'
all_files= [file for file in glob.glob(BASE_DIR+'/202[3|4]/*/*', recursive=True) if os.path.isfile(file)] # 2023/4 fi
actual_file ='./processed/2025/mar/mar2025_base.csv' # select latest file for Actuals
```

In [2]:
```
## actual data for the period 2023/2024
actual=pandas.read_csv(actual_file)
actual['ds']=pandas.to_datetime(actual['ds'])
actual=actual[(actual['forecast']==0) & (actual['ds'].dt.year.isin([2023,2024])) ].rename(columns={'values':'actual'}
## load forecasts
forecast=pandas.DataFrame()
for file in all_files:
    day_of_forecast= datetime.datetime.strptime(os.path.basename(file).split('_')[0],'%b%Y')
    day_of_forecast= str(day_of_forecast.year)+'-'+str(day_of_forecast.month).zfill(2)+'-'+str(day_of_forecast.day).z
    tmp=pandas.read_csv(file)
    tmp=tmp[tmp['forecast']==1]             # forecast data only
    tmp['date_of_forecast']=day_of_forecast # date on which forecast was done
    tmp.loc[:,'forecast_horizon'] = tmp['forecast'].expanding().sum()
    tmp.drop(columns=['forecast'],inplace=True)
    forecast=pandas.concat([forecast,tmp], axis=0)
forecast['ds']=pandas.to_datetime(forecast['ds'])
forecast['date_of_forecast']=pandas.to_datetime(forecast['date_of_forecast'])
forecast.rename(columns={'values':'forecast'},inplace=True)
```

1. Q: Analyze the forecast variance over time.

Solution: Line plot "variance of forecast" against "forecast horizon" (1months, 2months, 3months ...)
X-axis → Month of Forecast
Y-axis → Forecast mean Variance
It is expected that variance will increase with forecasting horizon

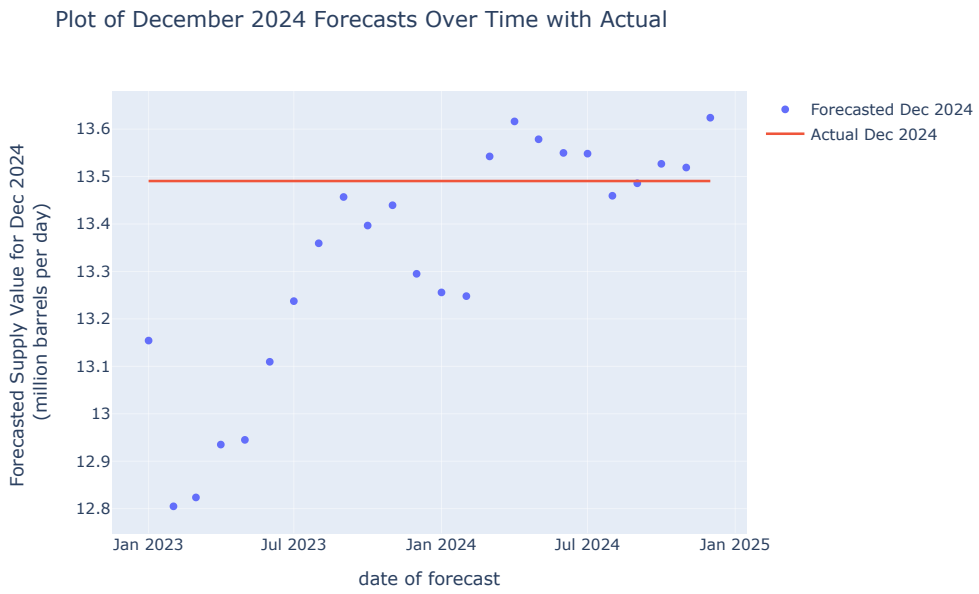
In [3]:
```
var=forecast.groupby('forecast_horizon').var().reset_index().rename(columns={'forecast':'variance'})
var=var[var['forecast_horizon']<13] # max forecast horizon of 1 year

fig =px.line(var, x='forecast_horizon', y='variance',title='Forecast Variance over forecasted horizon')\
.update_layout(xaxis_title="Forecast Horizon (in Months)", yaxis_title="Supply Forecast Variance <br>(million barrels
fig.show()
```



2. Q: What was the monthly deviation for the production estimate for December 2024 (forecast vs. actual expected today) over 2023 and 2024?

Solution: Line Plot of December 2024 Forecasts Over Time with Actual
X-axis → Forecast Month (Jan 2023 to Dec 2024)
Y-axis → Forecasted Value for Dec 2024
This shows how the forecast for December 2024 evolved each month.

In [4]:
```
tmp=forecast[(forecast['ds']=='2024-12-01')]
act_dec_2024=actual[actual['ds']=='2024-12-01']['actual'].values[0]
fig= go.Figure()
fig.update_layout(title='Plot of December 2024 Forecasts Over Time with Actual')
fig.add_trace(go.Scatter(x=tmp["date_of_forecast"], y=tmp['forecast'],mode='markers',name='Forecasted Dec 2024'))
fig.add_trace(go.Scatter(x=[tmp['date_of_forecast'].min(), tmp['date_of_forecast'].max()], y=[act_dec_2024, act_dec_2
fig.update_layout(xaxis_title="date of forecast", yaxis_title="Forecasted Supply Value for Dec 2024 <br>(million barr
fig.show()
```



3. Q: What are the range of deviations? (Dec 2024)

Solution: Range of deviation for Dec 2024 forecast
Box Plot of Forecast Errors for All Months
X-axis → Forecast Horizon (1M, 2M, … 12M ahead)
Y-axis → Forecast Error (Actual - Forecast)

In [5]:
```
tmp=forecast[(forecast['ds']=='2024-12-01')]
print("Range of deviation for dec 2024 forecast")
print('Min: ',tmp['forecast'].min())
print('Max: ',tmp['forecast'].max())
print('Mean: ',tmp['forecast'].mean())
print('Median: ',tmp['forecast'].median())
print('Actual: ',act_dec_2024)
```

```
Range of deviation for dec 2024 forecast
Min:  12.80489
Max:  13.62401
Mean:  13.329642916666666
Median:  13.41812
Actual:  13.490543
```

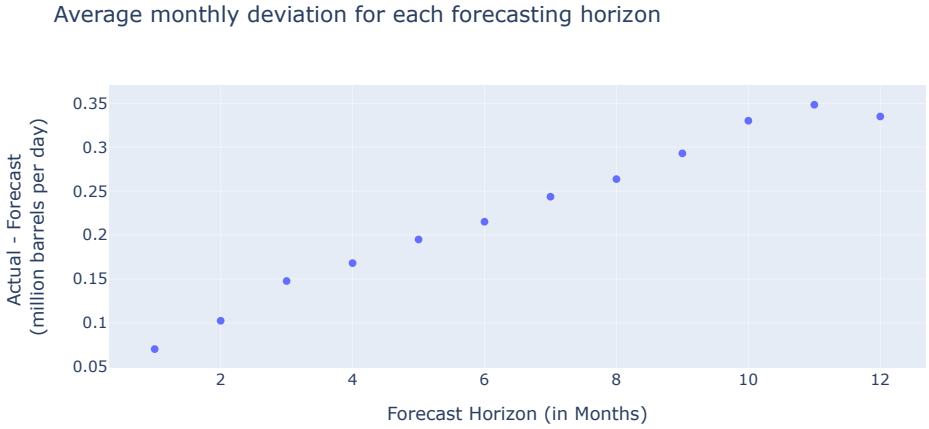
4. Q What is the average monthly deviation?

Assuming: the ask is to find average monthly deviation for one month horizon forecast Vs actual value.
Solution: Mean error for the Next month prediction.

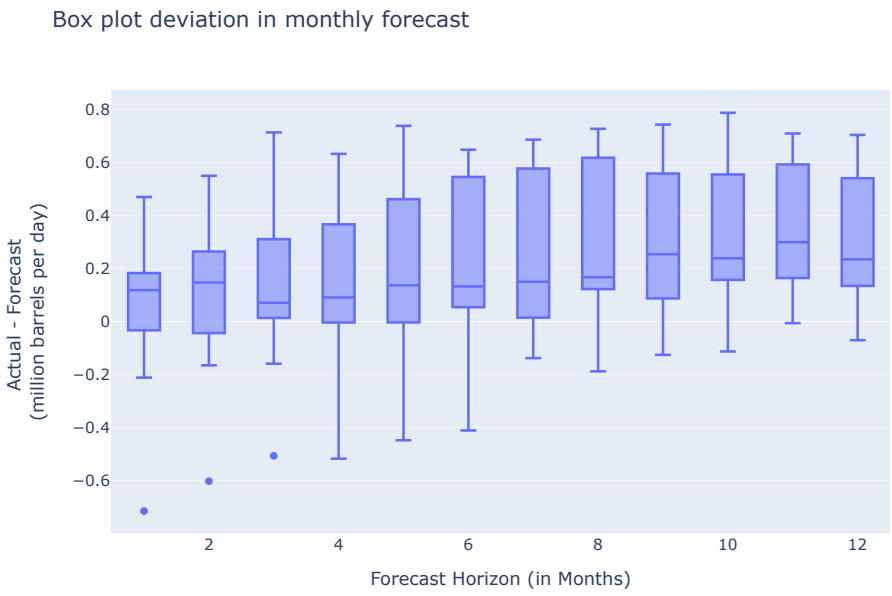
In [6]:
```
forecast= forecast[forecast['ds']<'2025-01-01']
tmp= forecast.merge(actual, how='left',on='ds')
tmp=tmp[tmp['forecast_horizon']<13]
tmp['deviation'] = tmp['actual'] - tmp['forecast']
tmp = tmp.groupby(['forecast_horizon'])['deviation'].mean().reset_index()
```

In [7]:
```
 Deviation:",tmp['deviation'].mean(),"(million barrels per day)" )

='forecast_horizon', y='deviation',title='Average monthly deviation for each forecasting horizon', height=400, width=8
s_title="Forecast Horizon (in Months)", yaxis_title=" Actual - Forecast <br>(million barrels per day)")
```

```
Average monthly Deviation: 0.22601765705840052 (million barrels per day)
```



In [8]:
```
tmp= forecast.merge(actual, how='left',on='ds')
tmp=tmp[tmp['forecast_horizon']<13]
tmp['deviation'] = tmp['actual'] - tmp['forecast']

fig=px.box(tmp,x="forecast_horizon", y='deviation',title='Box plot deviation in monthly forecast')
fig.update_layout(xaxis_title='Forecast Horizon (in Months)',yaxis_title='Actual - Forecast <br>(million barrels per
fig.show()
```

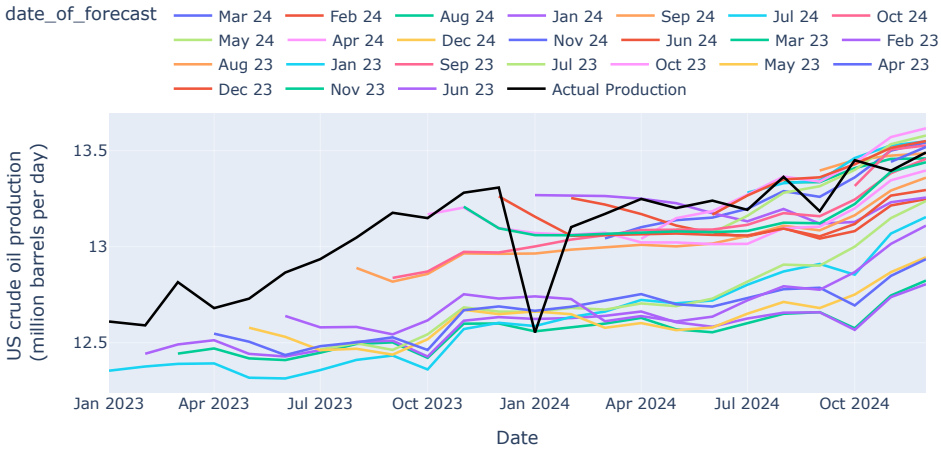


5. Q Plot the various forecast iterations on a chart.

Lets plot last two years of data each line represents start of forecast

In [9]:
```
forecast= forecast[forecast['ds']<'2025-01-01']
tmp= forecast.merge(actual, how='left',on='ds')
tmp['date_of_forecast']=pandas.to_datetime(tmp['date_of_forecast'])
tmp['date_of_forecast']=tmp['date_of_forecast'].dt.strftime("%b %y")

print('Line plot of each forecasting iteration \n ---------------------------------------')
fig=px.line(tmp, x='ds', y='forecast',color='date_of_forecast',height=400,width=800)
fig.add_trace(go.Scatter(x=actual['ds'],y=actual['actual'],mode='lines',line_color='black', name='Actual Production')
fig.update_layout(xaxis_title='Date',yaxis_title='US crude oil production <br>(million barrels per day)')
fig.update_layout(legend=dict(    orientation="h",    yanchor="bottom",
    y=1.02,    xanchor="right",    x=1 ))
fig.show()
```

```
Line plot of each forecasting iteration
 ---------------------------------------
```



In [10]:
```
# df= pandas.pivot_table(forecast, values=['forecast'] ,index = ['ds'], columns=['date_of_forecast'])
# df.index = df.index.strftime('%b %y') # Change index format
# temp_col=list(df.columns)             # Change column date format
# for ii in range(len(temp_col)):
#     temp_col[ii]=(temp_col[ii][0],temp_col[ii][1].strftime("%b %y"))
# df.columns= pandas.MultiIndex.from_tuples(temp_col, names=[None, 'date_of_forecast'])

# df.head(10)
```

In [11]:
```
df= pandas.pivot_table(forecast, values=['forecast'] ,index = ['date_of_forecast'], columns=['ds'])
df.index = df.index.strftime('%b %y') # Change index format
temp_col=list(df.columns)             # Change column date format
for ii in range(len(temp_col)):
    temp_col[ii]=(temp_col[ii][0],temp_col[ii][1].strftime("%b %y"))
df.columns= pandas.MultiIndex.from_tuples(temp_col, names=[None, 'ds'])
df.head(10)
```

Out[11]:

	forecast														
ds	Jan 23	Feb 23	Mar 23	Apr 23	May 23	Jun 23	Jul 23	Aug 23	Sep 23	Oct 23	...	Mar 24	Apr 24	May 24	Jun 24
date_of_forecast															
Jan 23	12.35465	12.37652	12.39013	12.39254	12.31802	12.31431	12.35787	12.41075	12.43362	12.36115	...	12.66222	12.72288	12.70548	12.71987
Feb 23	NaN	12.44287	12.49125	12.51323	12.44225	12.42862	12.46140	12.50467	12.51103	12.42842	...	12.64007	12.66266	12.60659	12.58324
Mar 23	NaN	NaN	12.44360	12.47030	12.41873	12.40987	12.44831	12.49393	12.50211	12.42159	...	12.59987	12.62953	12.56962	12.55461
Apr 23	NaN	NaN	NaN	12.54748	12.50557	12.43605	12.48220	12.50211	12.52837	12.46255	...	12.71840	12.75319	12.70042	12.68838
May 23	NaN	NaN	NaN	NaN	12.57830	12.53083	12.46168	12.46869	12.43823	12.51816	...	12.57789	12.60252	12.56572	12.57936
Jun 23	NaN	NaN	NaN	NaN	NaN	12.63946	12.58028	12.58248	12.54381	12.61657	...	12.61201	12.63899	12.61096	12.63583
Jul 23	NaN	NaN	NaN	NaN	NaN	NaN	12.46530	12.49745	12.46286	12.54392	...	12.67254	12.70474	12.69047	12.72971
Aug 23	NaN	NaN	NaN	NaN	NaN	NaN	NaN	12.88938	12.81814	12.85841	...	12.99501	13.00983	13.00167	13.01633
Sep 23	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	12.83766	12.87106	...	13.05979	13.08677	13.08859	13.08921
Oct 23	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	13.16856	...	13.07302	13.02235	13.02221	13.01364

10 rows × 24 columns

In [12]:
```
df.to_csv('summary_table.csv')
```