In [1]:
```
import pandas as pd
import numpy as np
import plotly.express as px
import plotly.graph_objects as go
pd.set_option('max_columns', None)
```

In this notebook, we will process data on connected peers obtained from monitoring the nodes. We first

1 IPFS

We want to know if the peers are consistently available throughout the day, or if the connected peers frequently change. We create the following graph similar to a heat map, indicating whether a peer is connected or not.

1205_peer_ipfs.txt:

```
[2022-12-05 00:00:01] /ip4/107.191.62.20/udp/4001/quic/p2p/12D3KooWJecJtRazm4ayfcg32Ks5LGi8JqMFzCfovmBVyWv3ZtoW
[2022-12-05 00:00:01] /ip4/116.202.229.43/udp/35784/quic/p2p/12D3KooWF73RLxETnMBCCfaJc9XujNxWdVGkjG1ayprffnGHhX5M
[2022-12-05 00:00:01] /ip4/129.159.35.103/tcp/4001/p2p/12D3KooWNTPtPGhjqJforqxfuUpqNVtNejkcXPfkrxTRP2W2gS74
...
```

In [2]:
```
df1 = pd.read_csv('1205_peer_ipfs.txt', sep=' ', header=None)
df1 = df1.rename(columns={0:"a",1:"b",2:"c"})
# get timestamp
df1['timestamp'] = df1['a'].map(str) +" "+ df1['b'].map(str)
df1['timestamp'] = df1['timestamp'].str.strip('[]')
df1['datetime'] = pd.to_datetime(df1['timestamp'])
# get peer
df1['peer'] = df1['c'].apply(lambda x: x.split('/')[-1])
df1['ip'] = df1['c'].apply(lambda x: x.split('/')[2])
df1 = df1.drop(['a','b','c','timestamp'],axis=1)
df1 = df1[['datetime','peer']].drop_duplicates()
df1.shape
```

Out[2]: (102818, 2)

In [3]:
```
df1.head()
```

Out[3]:

	datetime	peer
0	2022-12-05 00:00:01	12D3KooWJecJtRazm4ayfcg32Ks5LGi8JqMFzCfovmBVyW...
1	2022-12-05 00:00:01	12D3KooWF73RLxETnMBCCfaJc9XujNxWdVGkjG1ayprffn...
2	2022-12-05 00:00:01	12D3KooWNTPtPGhjqJforqxfuUpqNVtNejkcXPfkrxTRP2...
3	2022-12-05 00:00:01	12D3KooWFSJY3vgDusPw2fqkSJ47AHUtyfDseadSedM2kZ...
4	2022-12-05 00:00:01	12D3KooWMAVwRukMHAEnVv4MSEx7xbnwPXGQHx8foY6Cws...

In [4]:
```
len(df1['peer'].unique())
```

Out[4]: 7636

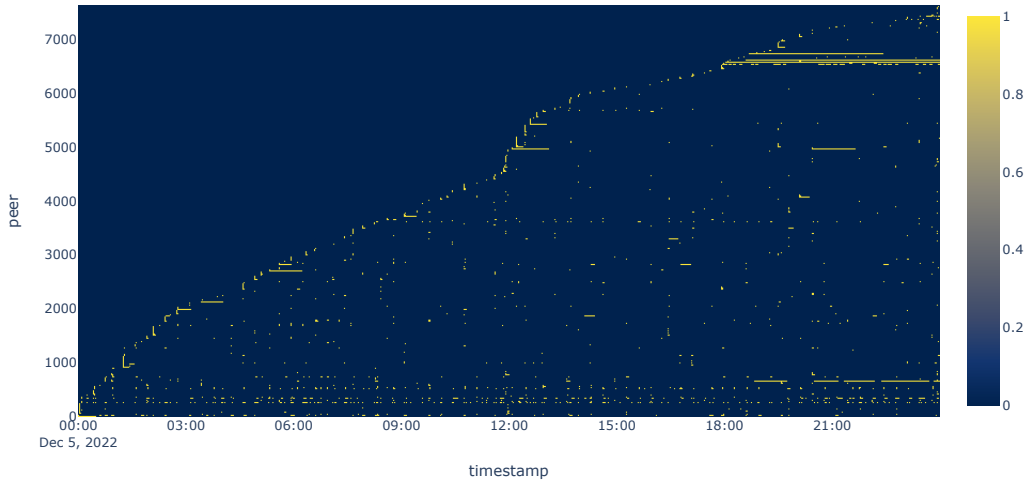
In [5]:
```
def getPivotDataframe(df):
    df1 = df.copy()
    df1['value'] = 1

    df2 = df1.pivot(index='datetime', columns='peer', values='value').T
    df2 = df2.fillna(0)
    # sort peers by order of occurance
    ll = list(df2.columns.strftime('%Y-%m-%d %H:%M:%S'))
    df2 = df2.sort_values(by=ll, axis=0, ascending=False)

    df2 = df2.reset_index()
    df2 = df2.drop(['peer'],axis=1)

    return df2
```

In [6]:
```
df2 = getPivotDataframe(df1)
df2.head()
```

Out[6]:

datetime	2022-12-05 00:00:01	2022-12-05 00:01:01	2022-12-05 00:02:01	2022-12-05 00:03:01	2022-12-05 00:04:01	2022-12-05 00:05:01	2022-12-05 00:06:01	2022-12-05 00:07:01	2022-12-05 00:08:01	2022-12-05 00:09:01	2022-12-05 00:10:01	2022-12-05 00:11:01	2022-12-05 00:12:01	2022-12-05 00:13:01	2022-12-05 00:14:01	2022-12-05 00:15:01	2022-12-05 00:16:01	2022-12-05 00:17:01	2022-12-05 00:18:01	2022-12-05 00:19:01
0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
1	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
2	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
3	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
4	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

In [7]:
```
fig = px.imshow(df2, color_continuous_scale="Cividis", origin='lower')
fig.update_xaxes(title="timestamp")
fig.update_yaxes(title="peer")
fig.show()
```

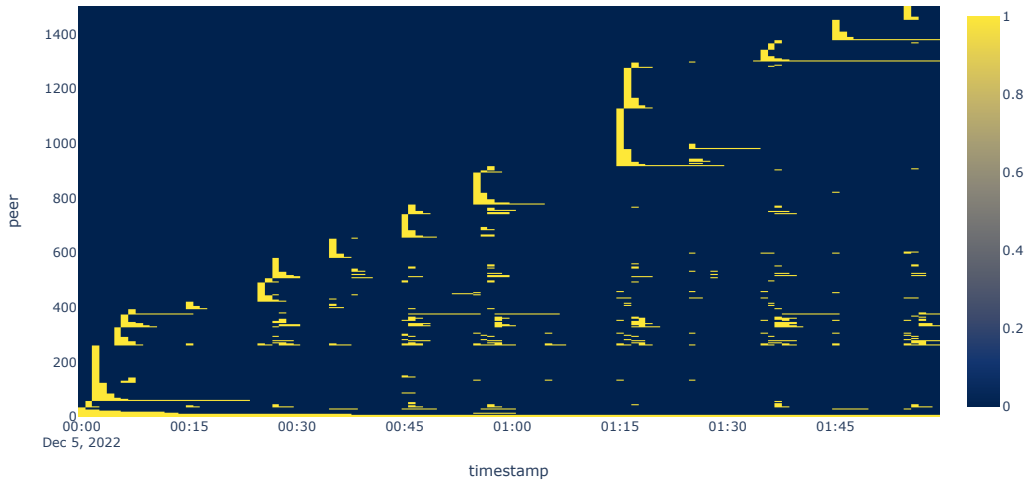


We zoom in to 2 hours to show the pattern more clearly.

In [8]:
```
df1_2 = df1[df1['datetime'].dt.strftime('%H:%M:%S').between('00:00:00','02:00:00')]

df2_2 = getPivotDataframe(df1_2)
df2_2.shape
```

Out[8]:
```
(1503, 120)
```

In [9]:
```
fig = px.imshow(df2_2, color_continuous_scale="Cividis", origin='lower')
fig.update_xaxes(title="timestamp")
fig.update_yaxes(title="peer")
fig.show()
```



2 Swarm

We do the same for swarm.

1205_peer_swarm.txt:

```
[2022-12-05 00:00:01] {"peers":[{"address":"002a5e864267f3478ecc1913a3a19ebdc84317af49e70bf96eed07f081c26e8a","fullNode":true},{"addres
s":"04e08d212a7fdda44e4143a733d9554fb456e8c895b7d11a242f9201be3ee012","fullNode":true},{"address":"f8572659a6cf70c7d98736983eb1efe9f7525ce5
13825f99f07355f333e23dac","fullNode":true}...]}
[2022-12-05 00:00:01]
[2022-12-05 00:01:01] {"peers":[{"address":"002a5e864267f3478ecc1913a3a19ebdc84317af49e70bf96eed07f081c26e8a","fullNode":true},{"addres
s":"04e08d212a7fdda44e4143a733d9554fb456e8c895b7d11a242f9201be3ee012","fullNode":true},{"address":"f8572659a6cf70c7d98736983eb1efe9f7525ce5
13825f99f07355f333e23dac","fullNode":true}...]}
...
```

In [10]:
```
df3 = pd.read_csv('1205_peer_swarm.txt', sep=' ', header=None)
df3 = df3.dropna()

df4 = pd.DataFrame(columns=['timestamp', 'peer'])

def func(a, i):
    a[i] = a[i].split('"')[3]

for index, row in df3.iterrows():
    timestamp = (row[0]+" "+row[1]).strip('[]')
    strs = row[2].split('{')[2:]
    list(map(lambda i:func(strs, i), range(0, len(strs))))
    df_temp = pd.DataFrame(strs)
    df_temp = df_temp.rename(columns={0: "peer"})
    df_temp['timestamp'] = timestamp
    df4 = pd.concat([df_temp, df4])

df4['datetime'] = pd.to_datetime(df4['timestamp'])
df4 = df4.drop(['timestamp'],axis=1)
df4.shape
```

Out[10]:
```
(187546, 2)
```

In [11]:
```
df4.head()
```

Out[11]:

	peer	datetime
0	044601f8bb98e4cfb6a394503020cafe010c6db0066f05...	2022-12-05 23:59:01
1	09d85feba81f7b7fde8858ada4c15c705dde7bcfb2a928...	2022-12-05 23:59:01
2	0c50592a898a3672d049a408893a961b7afe4f1308dcd7...	2022-12-05 23:59:01
3	11f028d16945b70ff6a677363ae13d05a073746ede3047...	2022-12-05 23:59:01
4	140771c0dc4451a833d0d777a0c4f67313c11bc3b5ab16...	2022-12-05 23:59:01

In [12]:
```
len(df4['peer'].unique())
```

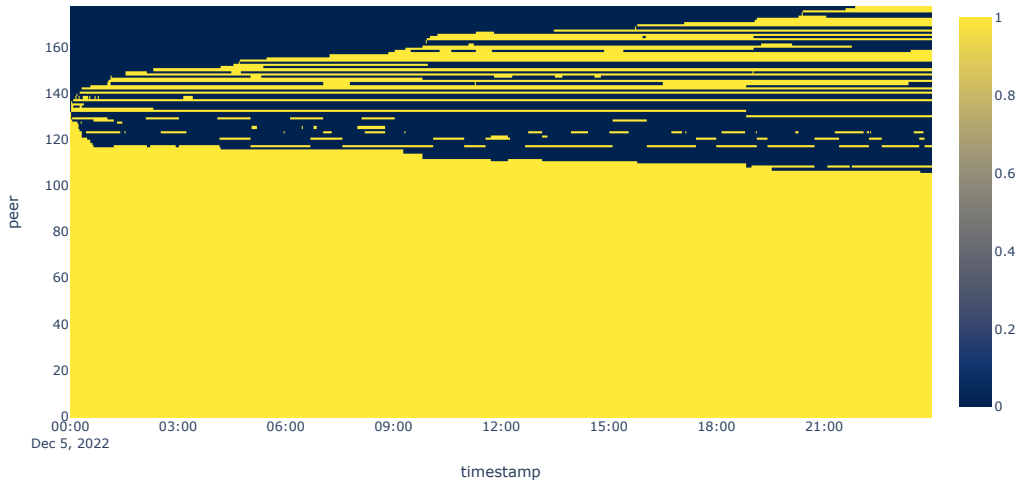
Out[12]:
```
178
```

In [13]:
```
df5 = getPivotDataframe(df4)
df5.head()
```

Out[13]:

datetime	2022-12-05 00:00:01	2022-12-05 00:01:01	2022-12-05 00:02:01	2022-12-05 00:03:01	2022-12-05 00:04:01	2022-12-05 00:05:01	2022-12-05 00:06:01	2022-12-05 00:07:01	2022-12-05 00:08:01	2022-12-05 00:09:01	2022-12-05 00:10:01	2022-12-05 00:11:01	2022-12-05 00:12:01	2022-12-05 00:13:01	2022-12-05 00:14:01	2022-12-05 00:15:01	2022-12-05 00:16:01	2022-12-05 00:17:01	2022-12-05 00:18:01	2022-12-05 00:19:01
0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
1	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
2	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
3	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
4	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

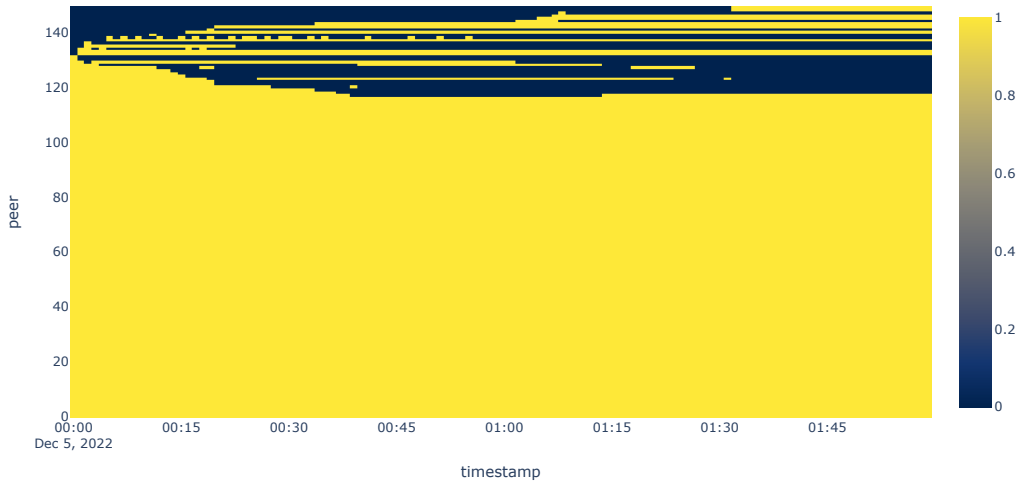
In [14]:
```
fig = px.imshow(df5, color_continuous_scale="Cividis", origin='lower')
fig.update_xaxes(title="timestamp")
fig.update_yaxes(title="peer")
fig.show()
```



In [15]:
```
df4_2 = df4[df4['datetime'].dt.strftime('%H:%M:%S').between('00:00:00','02:00:00')]

df5_2 = getPivotDataframe(df4_2)
df5_2.shape
```

Out[15]:
```
(150, 120)
```

In [16]:
```
fig = px.imshow(df5_2, color_continuous_scale="Cividis", origin='lower')
fig.update_xaxes(title="timestamp")
fig.update_yaxes(title="peer")
fig.show()
```



3 Compare

In [17]:
```
df6 = df1[['peer','datetime']].groupby('datetime').agg('count')
df6 = df6.rename(columns={'peer':'count'})
df6 = df6.reset_index()
df6['node'] = 'ipfs'
```

In [18]:
```
df6.describe()
```

Out[18]:

	count
count	1440.000000
mean	71.401389
std	62.559070
min	7.000000
25%	24.000000
50%	50.000000
75%	97.000000
max	447.000000

In [19]:
```
df7 = df4[['peer','datetime']].groupby('datetime').agg('count')
df7 = df7.rename(columns={'peer':'count'})
df7 = df7.reset_index()
df7['node'] = 'swarm'
```

In [20]:
```
df7.describe()
```

Out[20]:

	count
count	1440.000000
mean	130.240278
std	1.609316
min	126.000000
25%	129.000000
50%	130.000000
75%	131.000000
max	134.000000

In [21]:
```
df8 = pd.concat([df6, df7])
df8.head()
```

Out[21]:

	datetime	count	node
0	2022-12-05 00:00:01	37	ipfs
1	2022-12-05 00:01:01	50	ipfs
2	2022-12-05 00:02:01	232	ipfs
3	2022-12-05 00:03:01	89	ipfs
4	2022-12-05 00:04:01	47	ipfs

In [22]:
```
fig = px.line(df8, x='datetime', y='count', color='node')
fig.update_yaxes(title="number of connected peers")
fig.show()
```