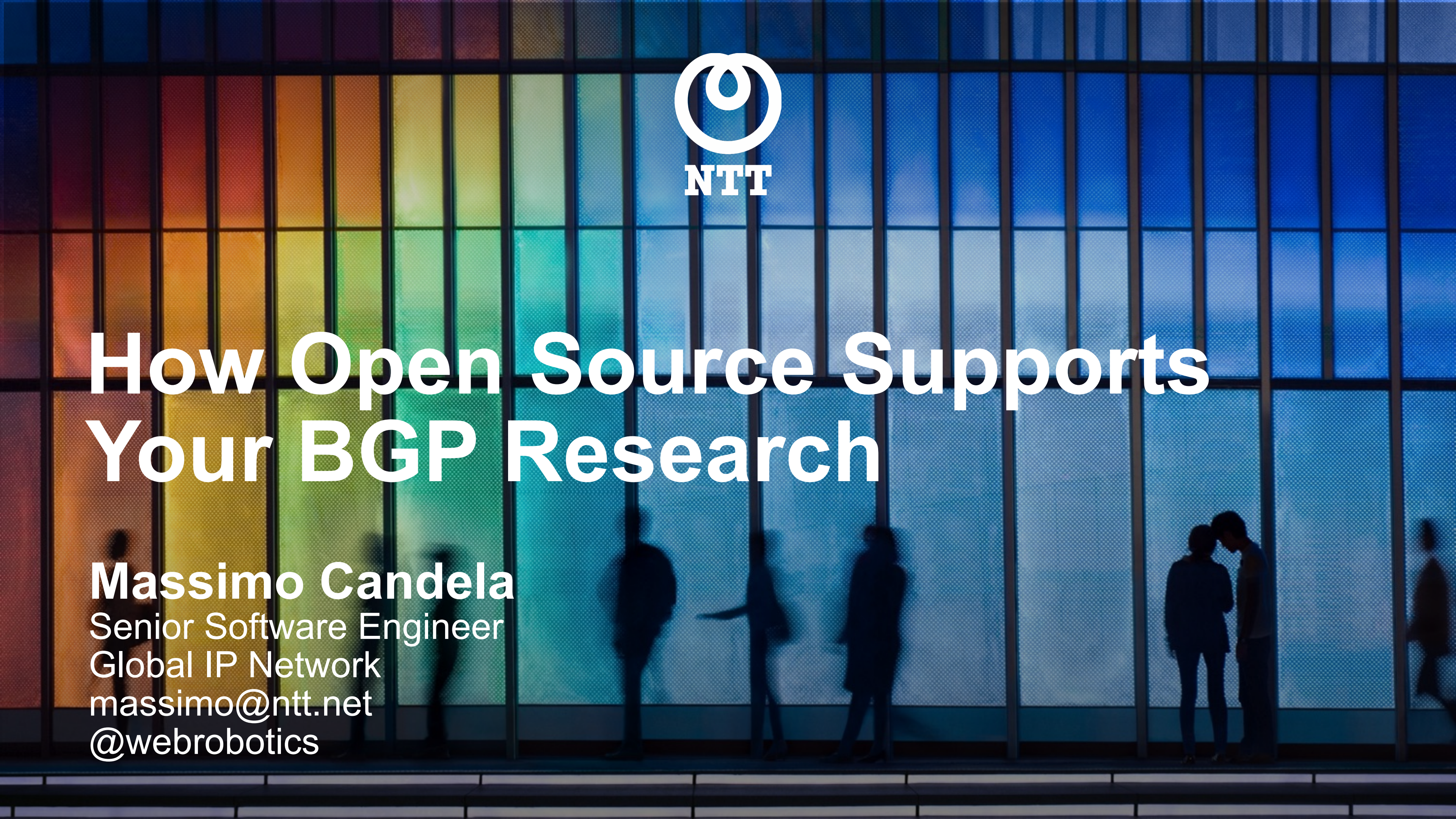


NTT

How Open Source Supports Your BGP Research

Massimo Candela
Senior Software Engineer
Global IP Network
massimo@ntt.net
@webrobotics

BGPalerter

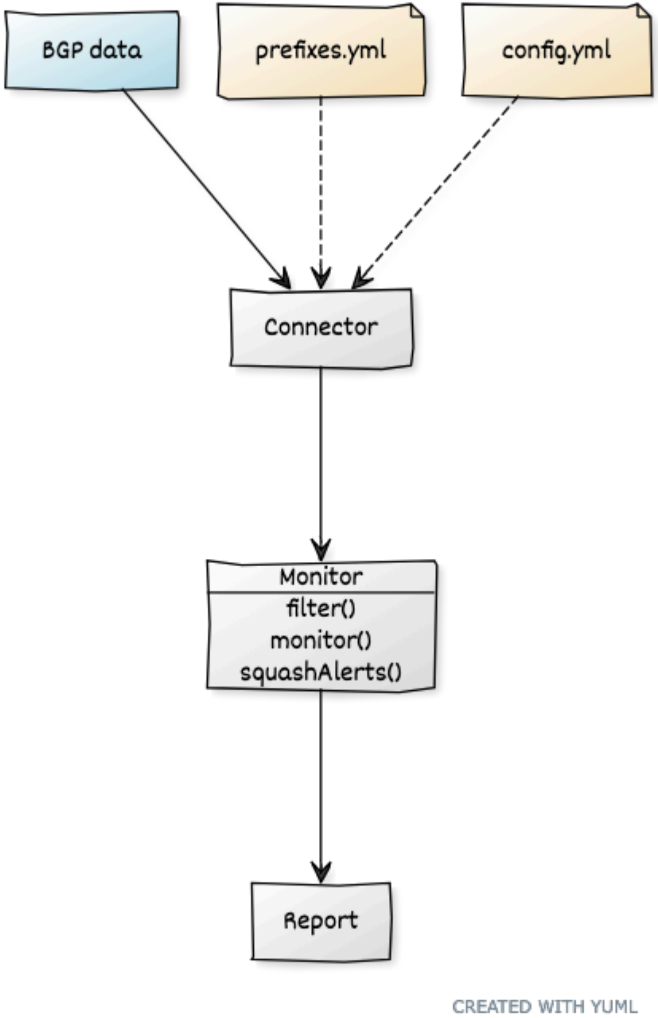
BGPalerter is a tool for analyzing streams of BGP data

- We developed it for monitoring NTT prefixes
  - hijacks, visibility loss, RPKI, and more
- We released it open-source (BSD-3-Clause)
  - https://github.com/nttgin/BGPalerter
- **It can be used for research!**

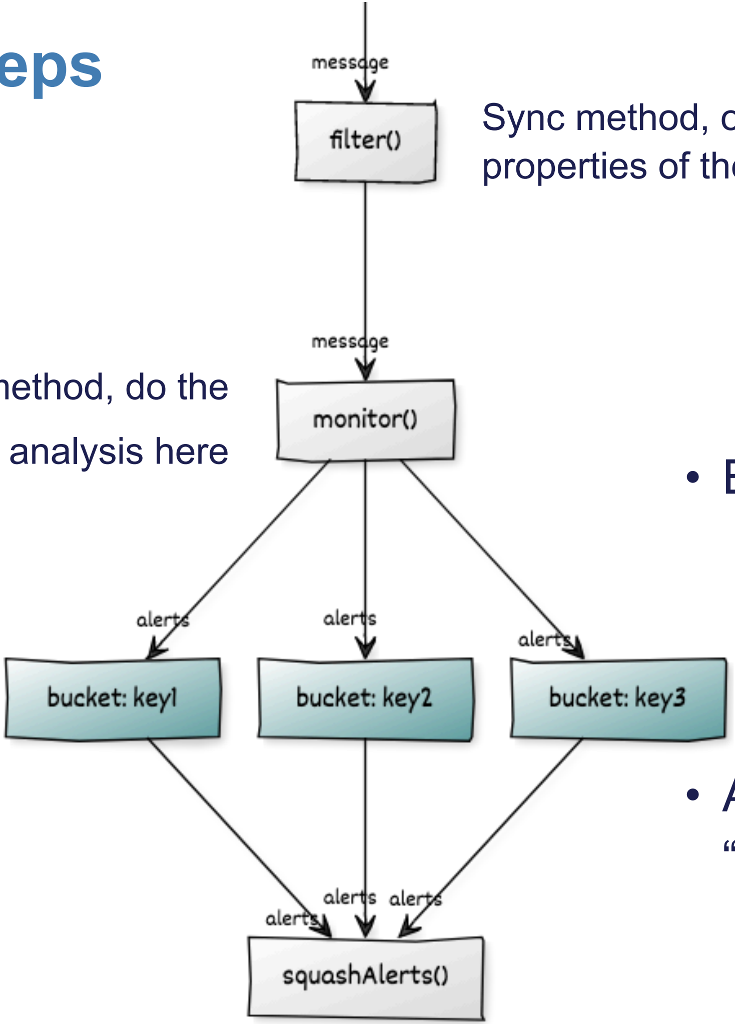
Example Use Case

- We want to monitor BGP announcements (*filter condition*)
- Detect bursts of RPKI invalid announcements (*monitor condition*)
  - We need to validate BGP announcements
- Every time a burst happen, we want to report the AS number (*group condition*), the number of RPKI invalids, and the BGP announcements
- We define as “burst” a number of invalid announcements >= 20 in 6 minutes (*squash condition*)

Pipeline


BGP data
prefixes.yml
config.yml
Connector
Monitor
filter()
monitor()
squashAlerts()
Report
CREATED WITH YUML

Monitor steps



Sync method, only check for intrinsic properties of the BGP message

Async method, do the complex analysis here

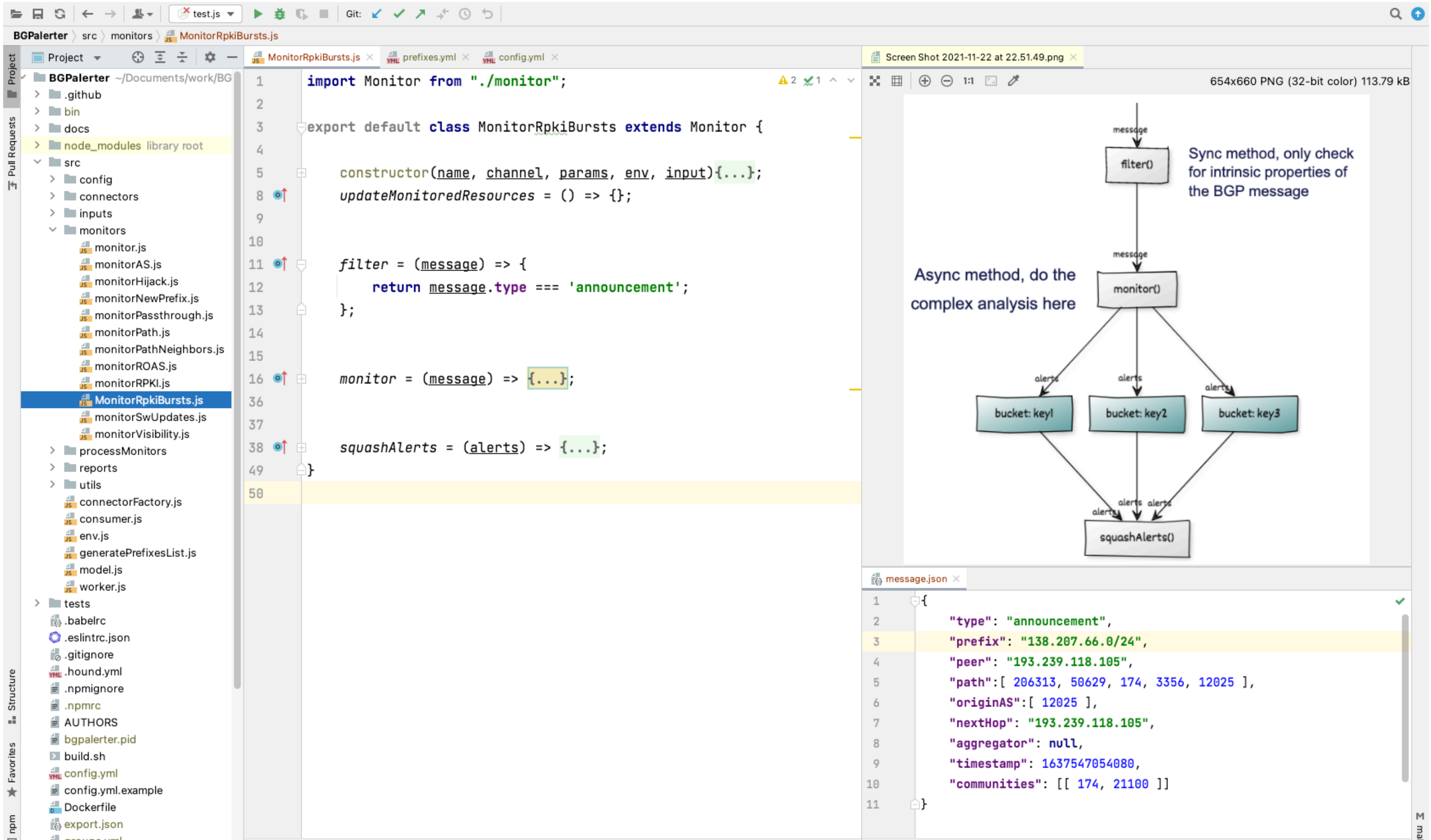
- Each alert is composed of:
  - Description of the verified monitored condition
  - BGP messages that verified the condition
- Alerts stay in the system for “fadeOffSeconds”

Example Use Case

- We want to monitor BGP announcements (***filter condition***)
- Detect bursts of RPKI invalid announcements (***monitor condition***)
  - We need to validate BGP announcements
- Every time a burst happen, we want to report the AS number (***group condition***), the number of RPKI invalids, and the BGP announcements
- We define as “burst” a number of invalid announcements >= 20 in 6 minutes (***squash condition***)

Example Use Case

- We want to monitor BGP announcements (***filter condition***)
- Detect bursts of RPKI invalid announcements (***monitor condition***)
  - We need to validate BGP announcements
- Every time a burst happen, we want to report the AS number (***group condition***), the number of RPKI invalids, and the BGP announcements
- We define as “burst” a number of invalid announcements >= 20 in 6 minutes (***squash condition***)



```
import Monitor from "./monitor";

export default class MonitorRpkiBursts extends Monitor {

    constructor(name, channel, params, env, input){...};
    updateMonitoredResources = () => {};


    filter = (message) => {
        return message.type === 'announcement';
    };


    monitor = (message) => {...};


    squashAlerts = (alerts) => {...};
}

```

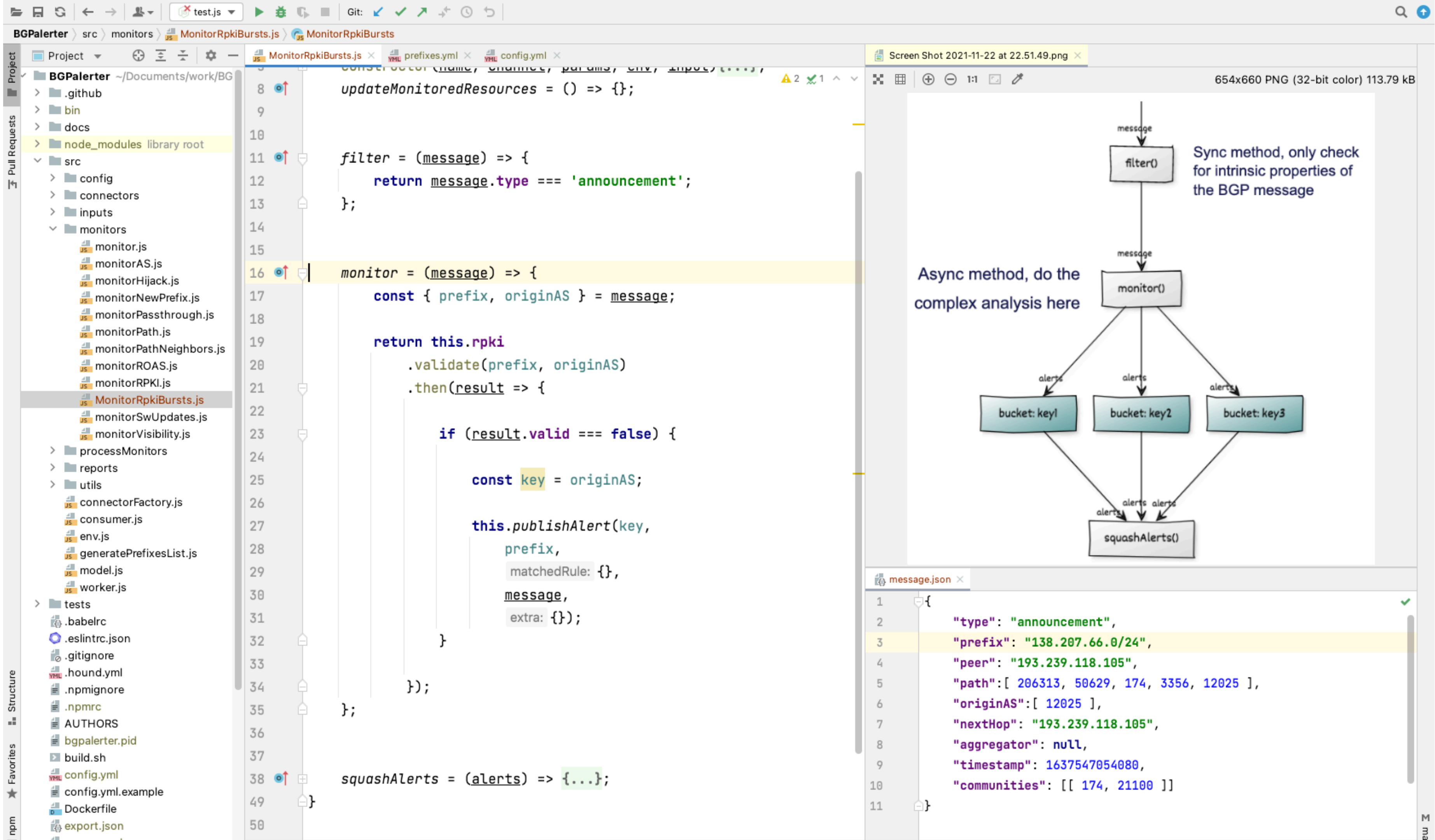
Sync method, only check for intrinsic properties of the BGP message

Async method, do the complex analysis here

```
{
    "type": "announcement",
    "prefix": "138.207.66.0/24",
    "peer": "193.239.118.105",
    "path":[ 206313, 50629, 174, 3356, 12025 ],
    "originAS":[ 12025 ],
    "nextHop": "193.239.118.105",
    "aggregator": null,
    "timestamp": 1637547054080,
    "communities": [[ 174, 21100 ]]
}
```

Example Use Case

- We want to monitor BGP announcements (***filter condition***)
- Detect bursts of RPKI invalid announcements (***monitor condition***)
  - We need to validate BGP announcements
- Every time a burst happen, we want to report the AS number (***group condition***), the number of RPKI invalids, and the BGP announcements
- We define as “burst” a number of invalid announcements >= 20 in 6 minutes (***squash condition***)



```
    constructor(name, channel, params, env, input){...};
    updateMonitoredResources = () => {};


    filter = (message) => {
        return message.type === 'announcement';
    };


    monitor = (message) => {
        const { prefix, originAS } = message;

        return this.rpki
            .validate(prefix, originAS)
            .then(result => {

                if (result.valid === false) {

                    const key = originAS;

                    this.publishAlert(key,
                        prefix,
                        matchedRule: {},
                        message,
                        extra: {});
                }

            });
    };


    squashAlerts = (alerts) => {...};
}
```

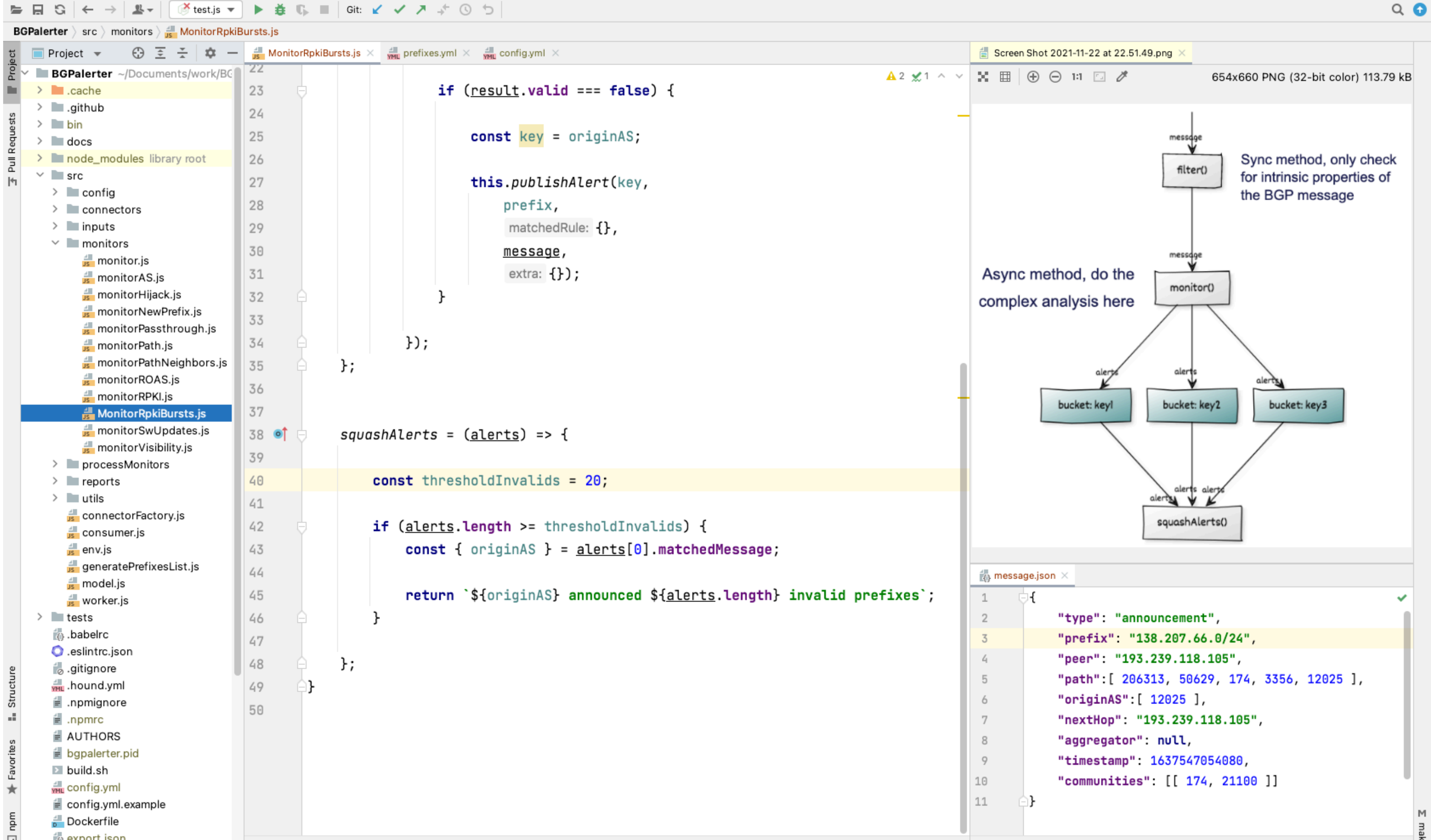
Sync method, only check for intrinsic properties of the BGP message

Async method, do the complex analysis here

```
{
    "type": "announcement",
    "prefix": "138.207.66.0/24",
    "peer": "193.239.118.105",
    "path":[ 206313, 50629, 174, 3356, 12025 ],
    "originAS":[ 12025 ],
    "nextHop": "193.239.118.105",
    "aggregator": null,
    "timestamp": 1637547054080,
    "communities": [[ 174, 21100 ]]
}
```

Example Use Case

- We want to monitor BGP announcements (***filter condition***)
- Detect bursts of RPKI invalid announcements (***monitor condition***)
    - We need to validate BGP announcements
- Every time a burst happen, we want to report the AS number (***group condition***), the number of RPKI invalids, and the BGP announcements
- We define as “burst” a number of invalid announcements >= 20 in 6 minutes (***squash condition***)

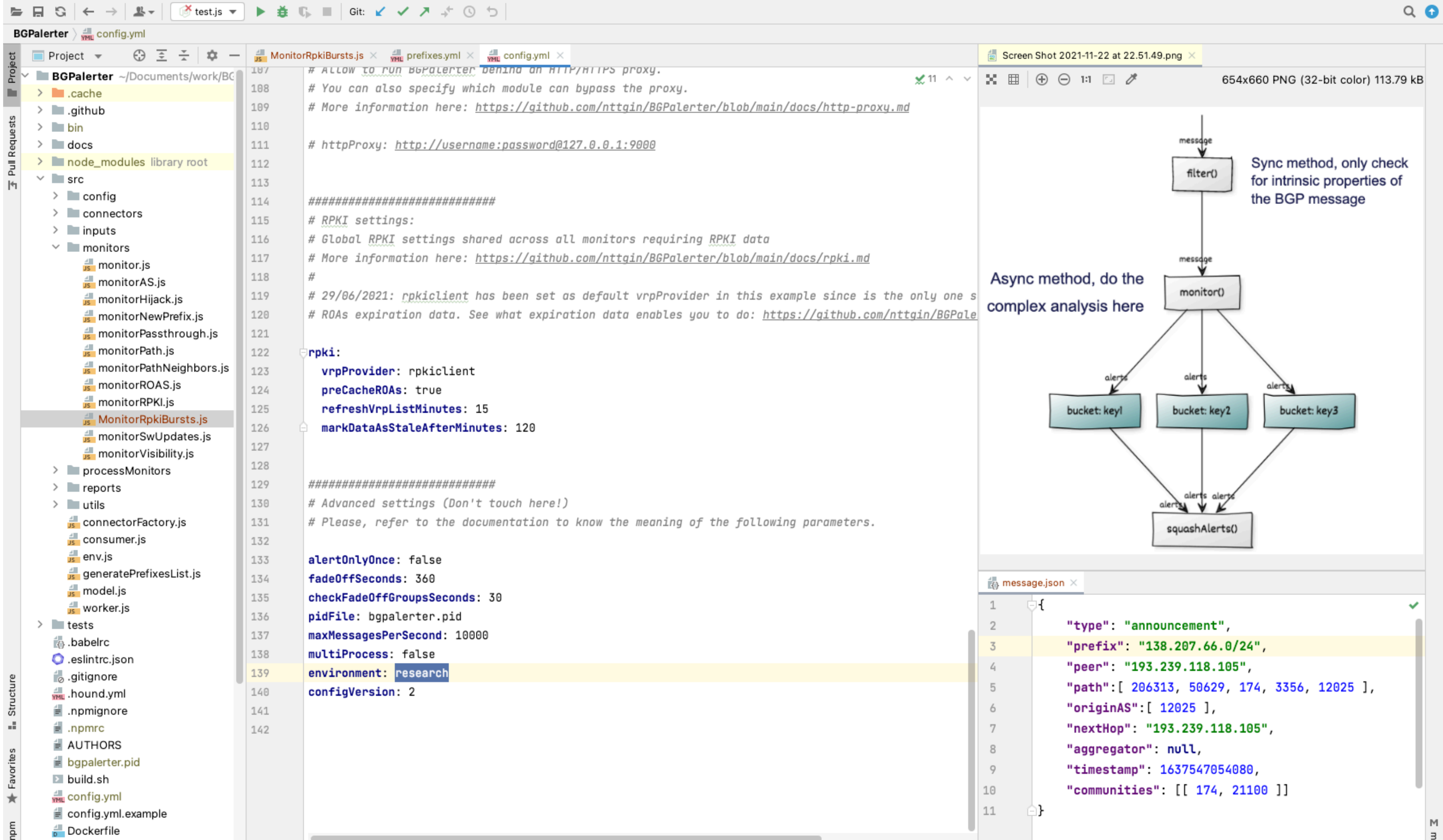


```
                if (result.valid === false) {

                    const key = originAS;

                    this.publishAlert(key,
                        prefix,
                        matchedRule: {},
                        message,
                        extra: {});
                }

            });
    };


    squashAlerts = (alerts) => {

        const thresholdInvalids = 20;

        if (alerts.length >= thresholdInvalids) {
            const { originAS } = alerts[0].matchedMessage;

            return `${originAS} announced ${alerts.length} invalid prefixes`;
        }

    };
}
```

Sync method, only check for intrinsic properties of the BGP message

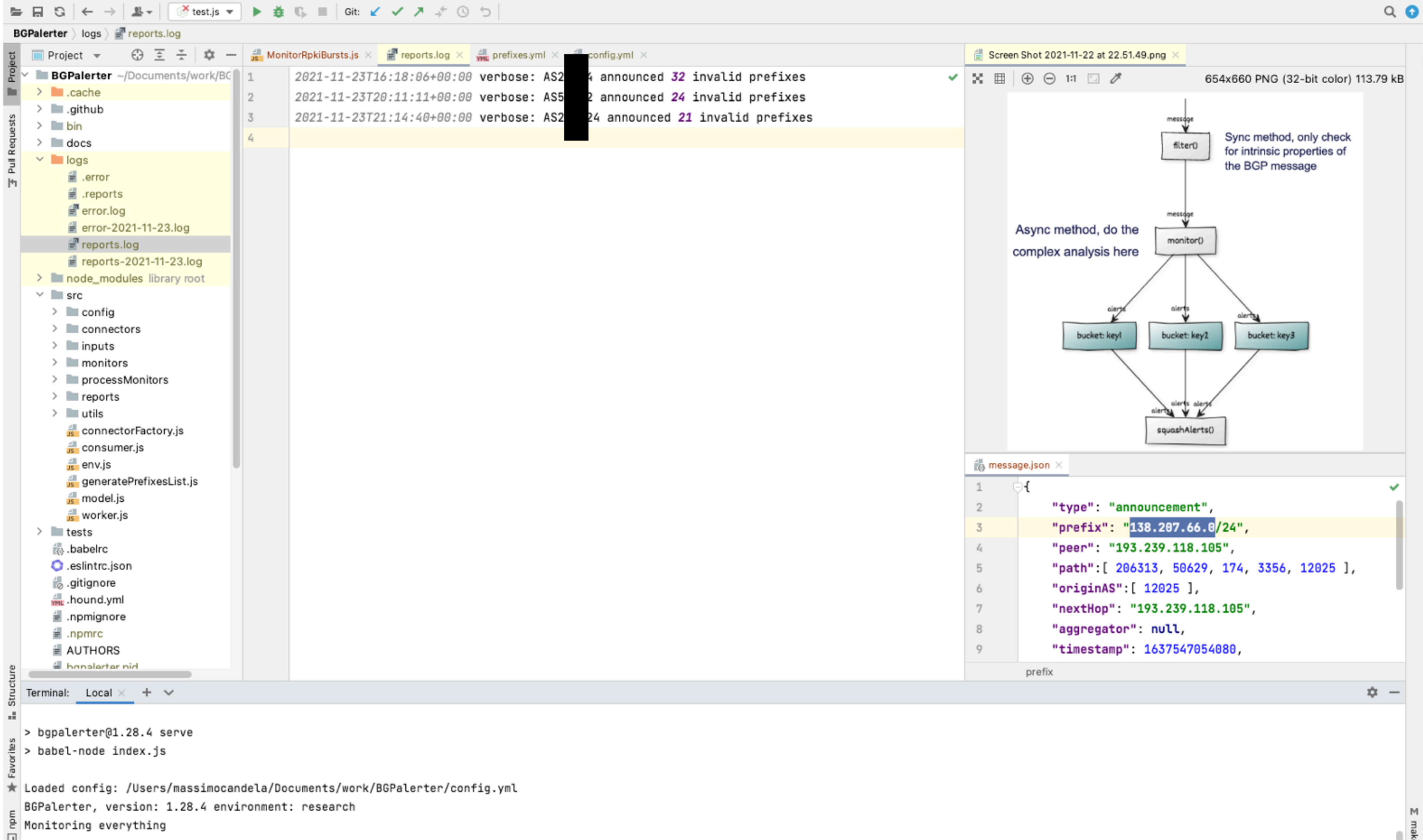
Async method, do the complex analysis here

```
{
    "type": "announcement",
    "prefix": "138.207.66.0/24",
    "peer": "193.239.118.105",
    "path":[ 206313, 50629, 174, 3356, 12025 ],
    "originAS":[ 12025 ],
    "nextHop": "193.239.118.105",
    "aggregator": null,
    "timestamp": 1637547054080,
    "communities": [[ 174, 21100 ]]
}
```



```
# Allow to run BGPalerter behind an HTTP/HTTPS proxy.
# You can also specify which module can bypass the proxy.
# More information here: https://github.com/nttgin/BGPalerter/blob/main/docs/http-proxy.md

# httpProxy: http://username:password@127.0.0.1:9000


############################
# RPKI settings:
# Global RPKI settings shared across all monitors requiring RPKI data
# More information here: https://github.com/nttgin/BGPalerter/blob/main/docs/rpki.md
#
# 29/06/2021: rpkiclient has been set as default vrpProvider in this example since is the only one s
# ROAs expiration data. See what expiration data enables you to do: https://github.com/nttgin/BGPale

rpki:
  vrpProvider: rpkiclient
  preCacheROAs: true
  refreshVrpListMinutes: 15
  markDataAsStaleAfterMinutes: 120


############################
# Advanced settings (Don't touch here!)
# Please, refer to the documentation to know the meaning of the following parameters.

alertOnlyOnce: false
fadeOffSeconds: 360
checkFadeOffGroupsSeconds: 30
pidFile: bgpalerter.pid
maxMessagesPerSecond: 10000
multiProcess: false
environment: research
configVersion: 2


```

```
{
    "type": "announcement",
    "prefix": "138.207.66.0/24",
    "peer": "193.239.118.105",
    "path":[ 206313, 50629, 174, 3356, 12025 ],
    "originAS":[ 12025 ],
    "nextHop": "193.239.118.105",
    "aggregator": null,
    "timestamp": 1637547054080,
    "communities": [[ 174, 21100 ]]
}
```



```
2021-11-23T16:18:06+00:00 verbose: AS2[redacted] announced 32 invalid prefixes
2021-11-23T20:11:11+00:00 verbose: AS5 announced 24 invalid prefixes
2021-11-23T21:14:40+00:00 verbose: AS2 announced 21 invalid prefixes
```

```
{
    "type": "announcement",
    "prefix": "138.207.66.0/24",
    "peer": "193.239.118.105",
    "path":[ 206313, 50629, 174, 3356, 12025 ],
    "originAS":[ 12025 ],
    "nextHop": "193.239.118.105",
    "aggregator": null,
    "timestamp": 1637547054080,
```

```
> bgpalerter@1.28.4 serve
> babel-node index.js

Loaded config: /Users/massimocandela/Documents/work/BGPalerter/config.yml
BGPalerter, version: 1.28.4 environment: research
Monitoring everything
```

Contribute!

- Source code on GitHub
  - https://github.com/nttgin/BGPalerter

Thank you.

Massimo Candela

Senior Software Engineer, Network Information Systems Development

Global IP Network

massimo@ntt.net

@webrobotics

www.gin.ntt.net

@GinNTTnet #globalipnetwork #AS2914