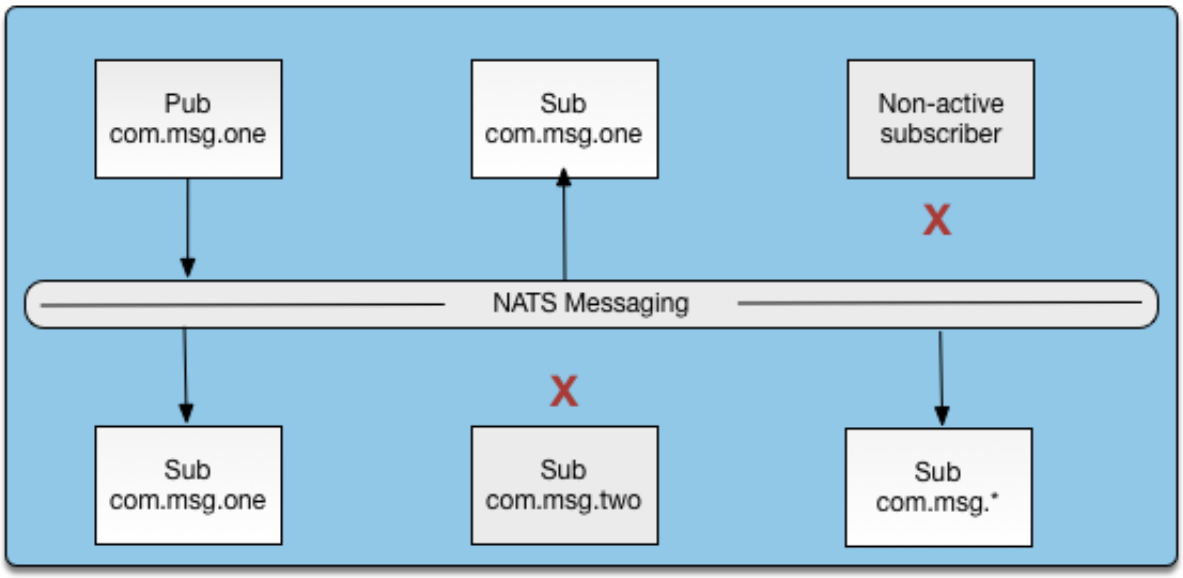
Explore NATS Pub Sub

NATS is a publish subscribe messaging system (/documentation/concepts/nats-pub-sub/). Subscribers listening on a subject name receive messages on that subject. If the subscriber is not actively listening on the subject, the message is not received. Subscribers can use the wildcard subjects `*` to match a single token to match the tail of a subject.



Prerequisites

- Set up your Go environment (/documentation/tutorials/go-install/)
- Installed the NATS server (/documentation/tutorials/gnatsd-install/)

Instructions

1. Start the NATS server.

```
gnatsd
```

When the server starts successfully, you will see the following messages:

```
[1] 2015/08/12 15:18:22.301550 [INF] Starting gnatsd version 0.6.4
[1] 2015/08/12 15:18:22.301762 [INF] Listening for client connectio
ns on 0.0.0.0:4222
[1] 2015/08/12 15:18:22.301769 [INF] gnatsd is ready
```

The NATS server listens for client connections on TCP Port 4222.

2. Start a shell or command prompt session.

You will use this session to run an example NATS client subscriber program.

3. CD to the example Go client directory.

```
cd $GOPATH/src/github.com/nats-io/nats/examples
```

4. Run the client subscriber program.

```
go run nats-sub.go <subject>
```

Where is a subject to listen on. A valid subject is a string that is unique in the system.

For example:

```
go run nats-sub.go msg.test
```

You should see the message: *Listening on [msg.test]*

5. Start another shell or command prompt session.

You will use this session to run a NATS publisher client.

6. CD to the examples directory.

```
cd $GOPATH/src/github.com/nats-io/nats/examples
```

7. Publish a NATS message.

```
go run nats-pub.go <subject> <"message”>
```

Where is the subject name and <“message”> is a message to publish.

For example:

```
go run nats-pub.go msg.test "NATS MESSAGE"
```

7. Verify message publication and receipt.

You should see that the publisher sends the message: *Published [msg.test] : ‘NATS MESSAGE’*

And that the subscriber receives the message: *[#1] Received on [msg.test]: ‘NATS MESSAGE’*

Note that if the receiver does not get the message, check that you are using the same subject name for the publisher and the subscriber.

8. Publish another message.

```
go run nats-pub.go msg.test "NATS MESSAGE 2"
```

You should see that the subscriber receive message 2. Note that the message count is incremented each time your subscribing client receives a message on that subject:

9. Start another shell or command prompt session.

You will use this session to run a second NATS subscriber.

10. CD to the examples directory.

```
cd $GOPATH/src/github.com/nats-io/nats/examples
```

11. Subscribe to the message.

```
go run nats-sub.go msg.test
```

12. Publish another message using the publisher client.

```
go run nats-pub.go msg.test "NATS MESSAGE 3"
```

Verify that both subscribing clients receive the message.

13. Start another shell or command prompt session.

You will use this session to run a third NATS subscriber.

14. CD to the examples directory.

```
cd $GOPATH/src/github.com/nats-io/nats/examples
```

15. Subscribe to a different message.

```
go run nats-sub.go msg.test.new
```

All the but last subscriber receives the message. Why? Because that subscriber is not listening on the message subject used by the publisher.

16. Update the last subscriber to use a wildcard.

NATS supports the use of wildcard characters for message subscribers. (You cannot publish a message using a wildcard subject.)

Change the last subscriber the listen on msg.* and run it:

```
go run nats-sub.go msg.*
```

17. Publish another message.

This time, all three subscribing clients should receive the message.