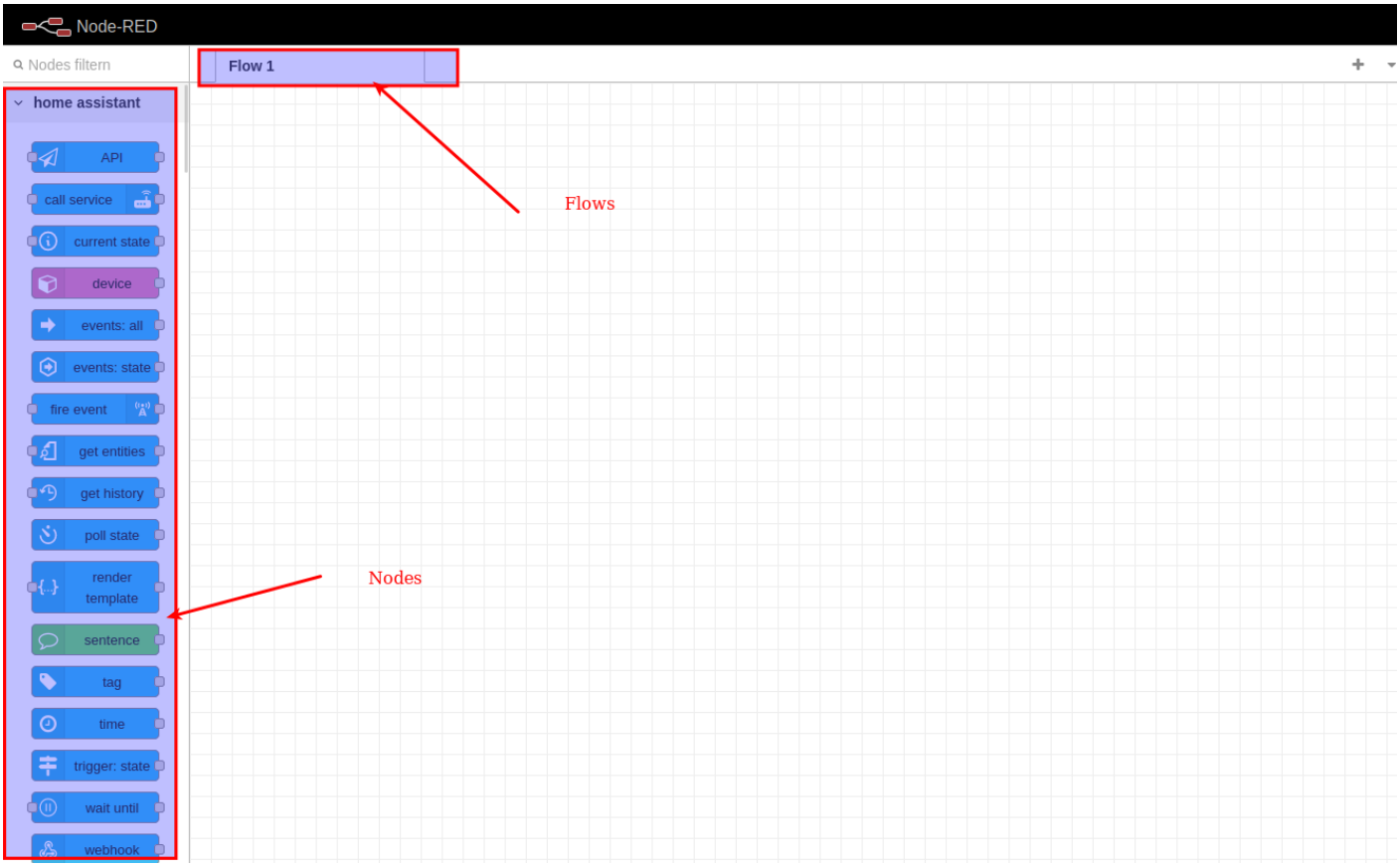
Grundübersicht

Flows und Nodes

Übersicht

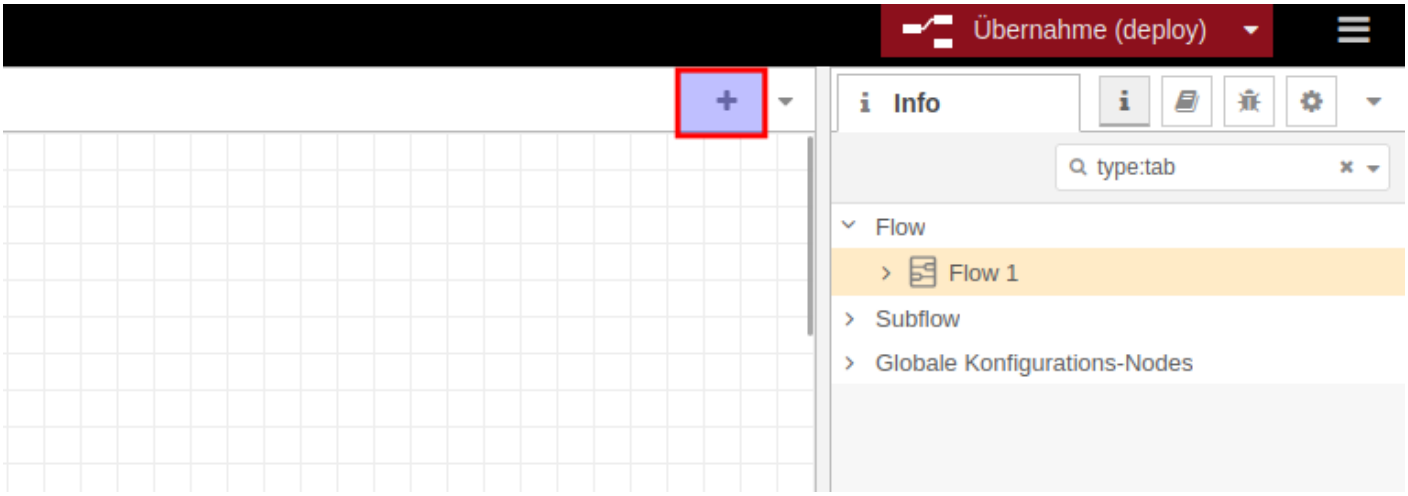
Flows sind die Seiten auf den Nodes plaziert werden. Mann kann mehrere Flows haben. Vergleichbar mit Szenen.
Somit wird es übersichtlicher. Man kann natürlich auch alles auf einen Flow packen. Aber der Übersichthalber kann man halt mehrere Flows anlegen.

Die Nodes sind auf der Linke Seite.
Nodes sind unsere Bausteine mit den wir unseren Flow bauen.

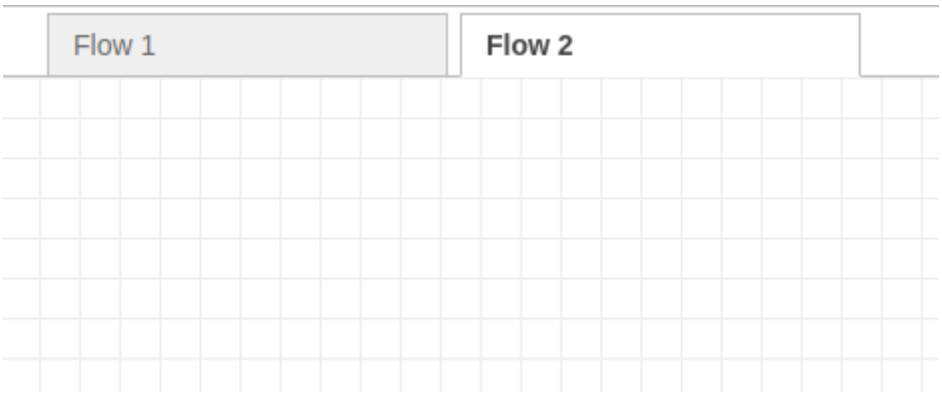


Flow hinzufügen

Um ein weiteren Flow hinzu zu fügen klicken wir auf das + Symbol



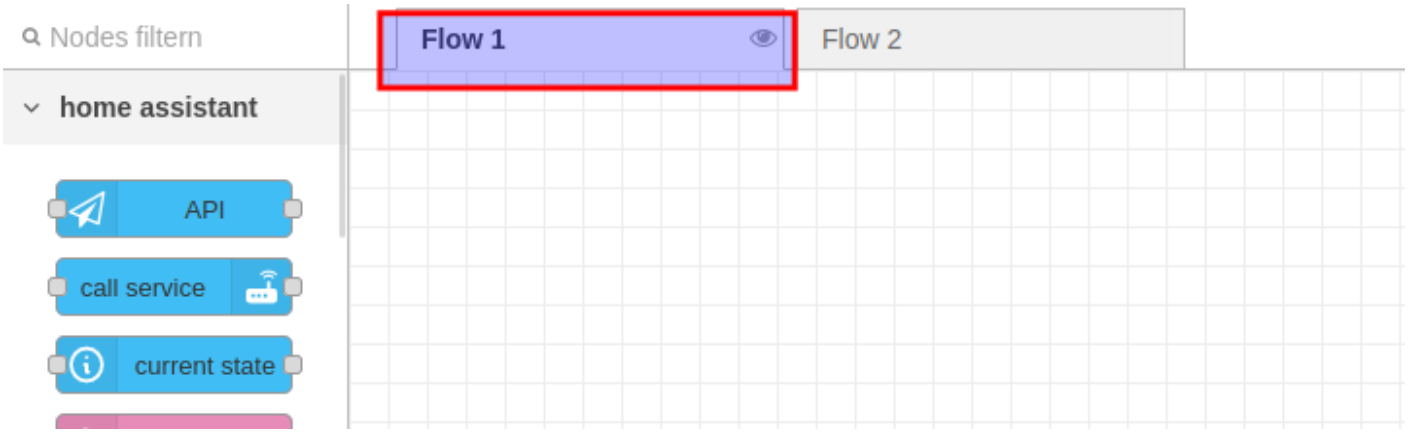
Nun haben wir einen zweiten Flow im Tab daneben.



Im Rechten Fenster sehen wir auch nochmal unsere Flows.

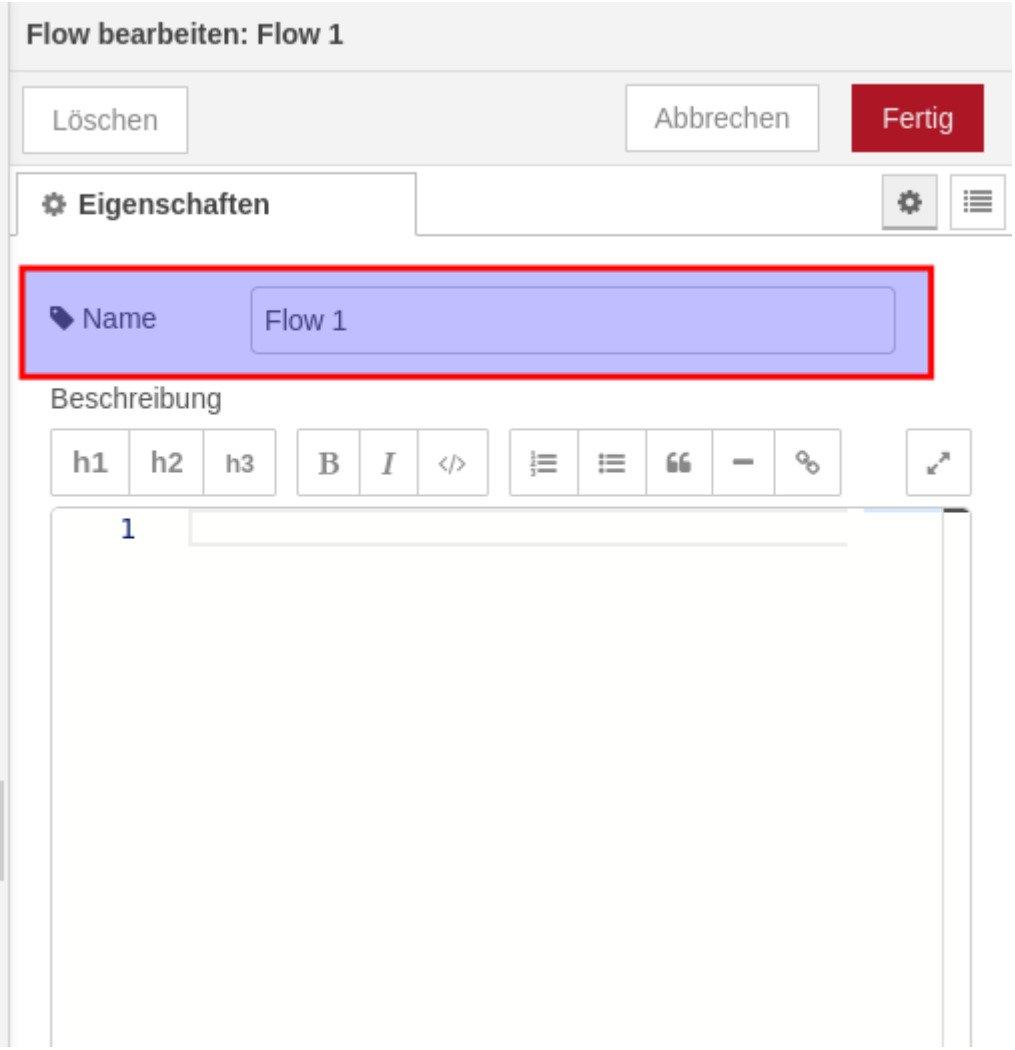
Flow umbenennen

Um einen Flow umzubenennen, doppelklick auf den Tab des Flows den man umbenennen möchte

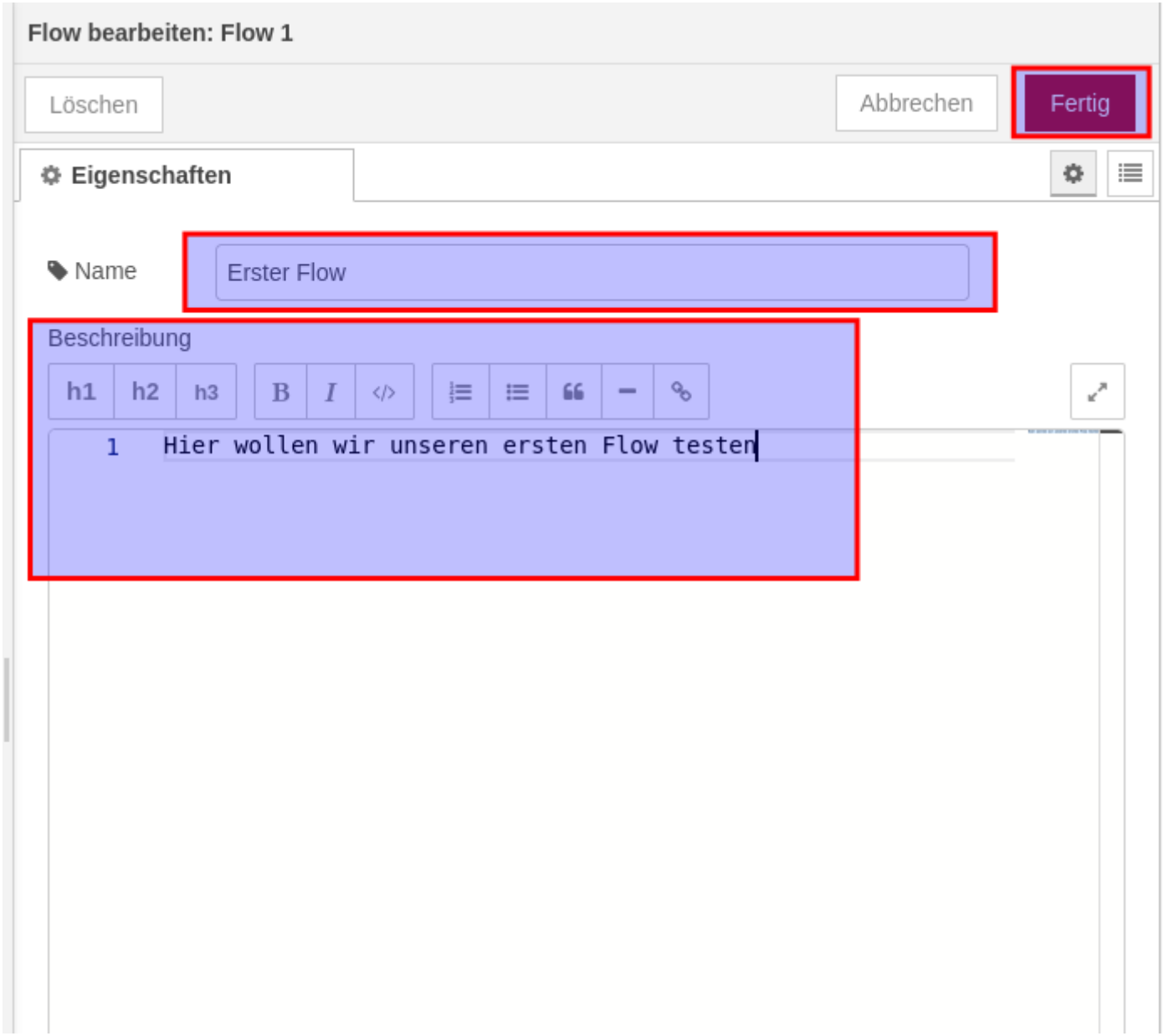


Nun dort den Text ändern. Wir haben auch die Möglichkeit eine Beschreibung hinzuzufügen. Darin kann man zum Beispiel noch mal seine Abläufe erläutern, damit man nach einem jahr auch

noch weiß was womit gemeint war, so ne Art Doku.

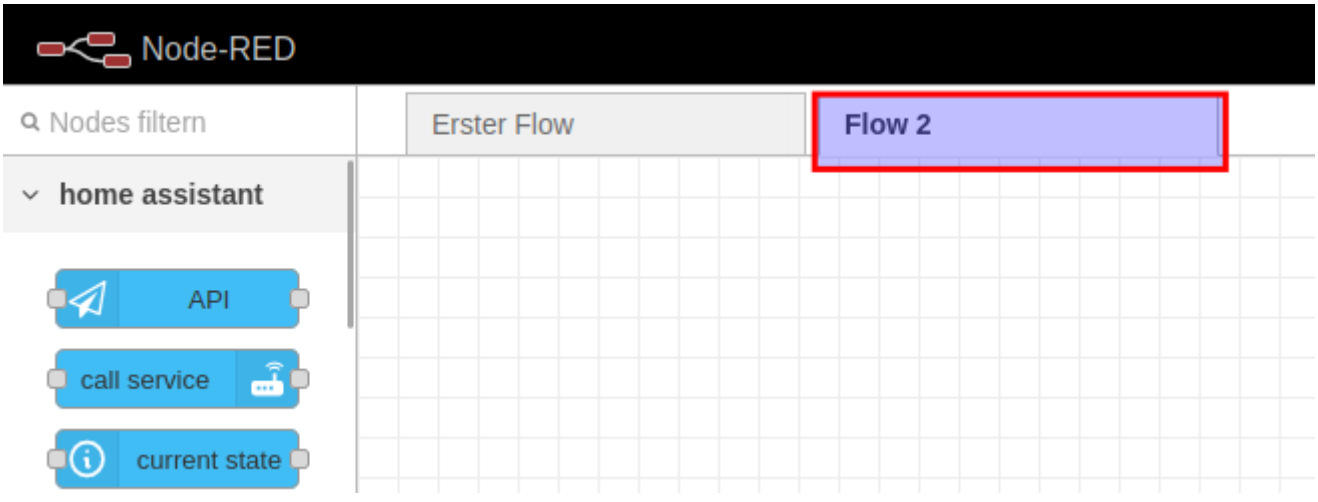


Geändert und eine Beschreibung eingefügt. Nun auf Fertig klicken.

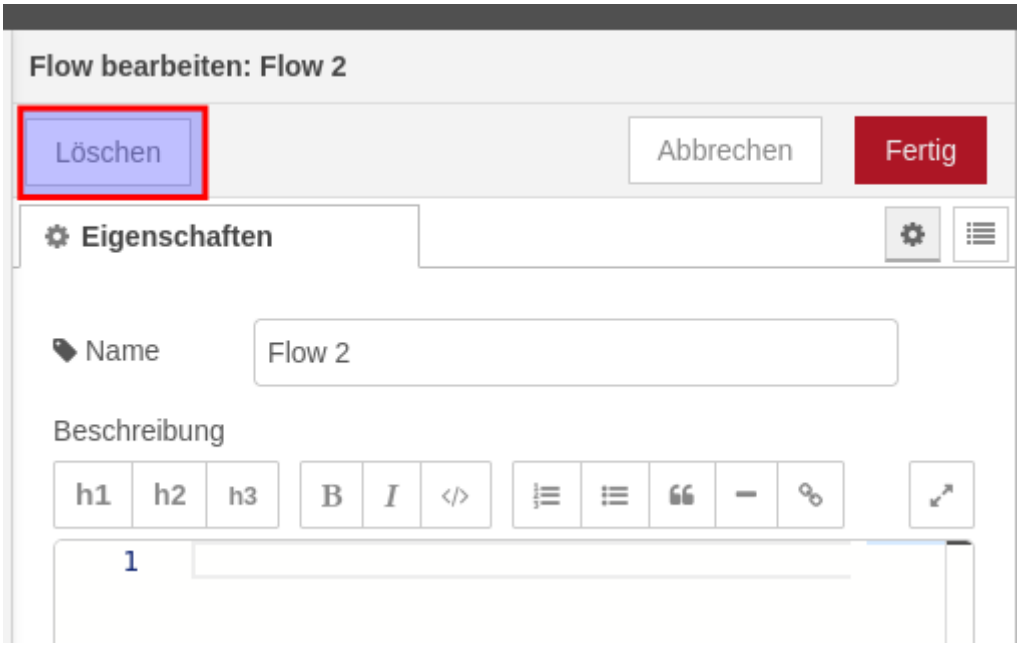


Flow löschen

Doppelklcik wieder auf den Reiter des zu löschenden Flows



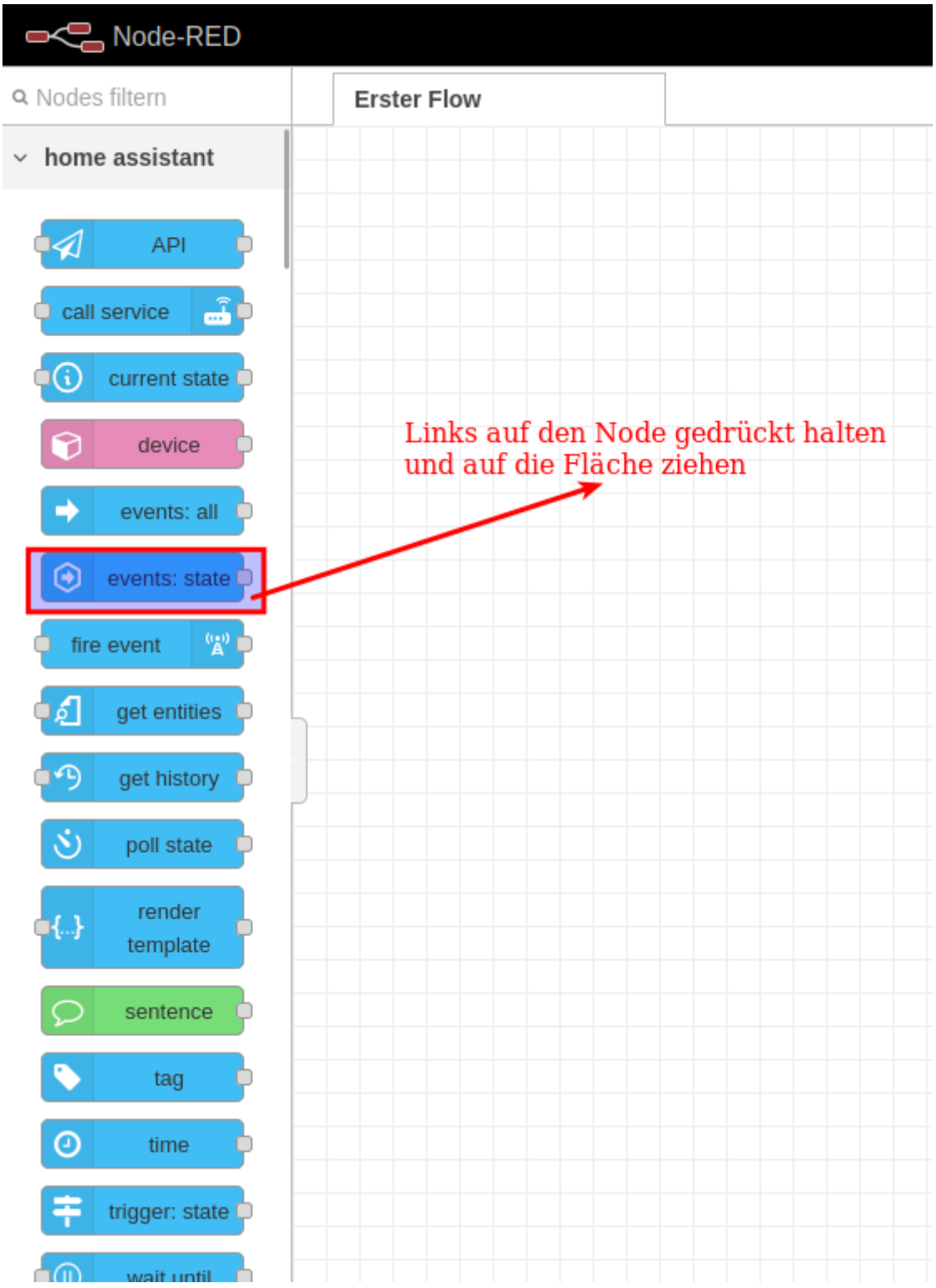
Und dort dann löschen anklicken



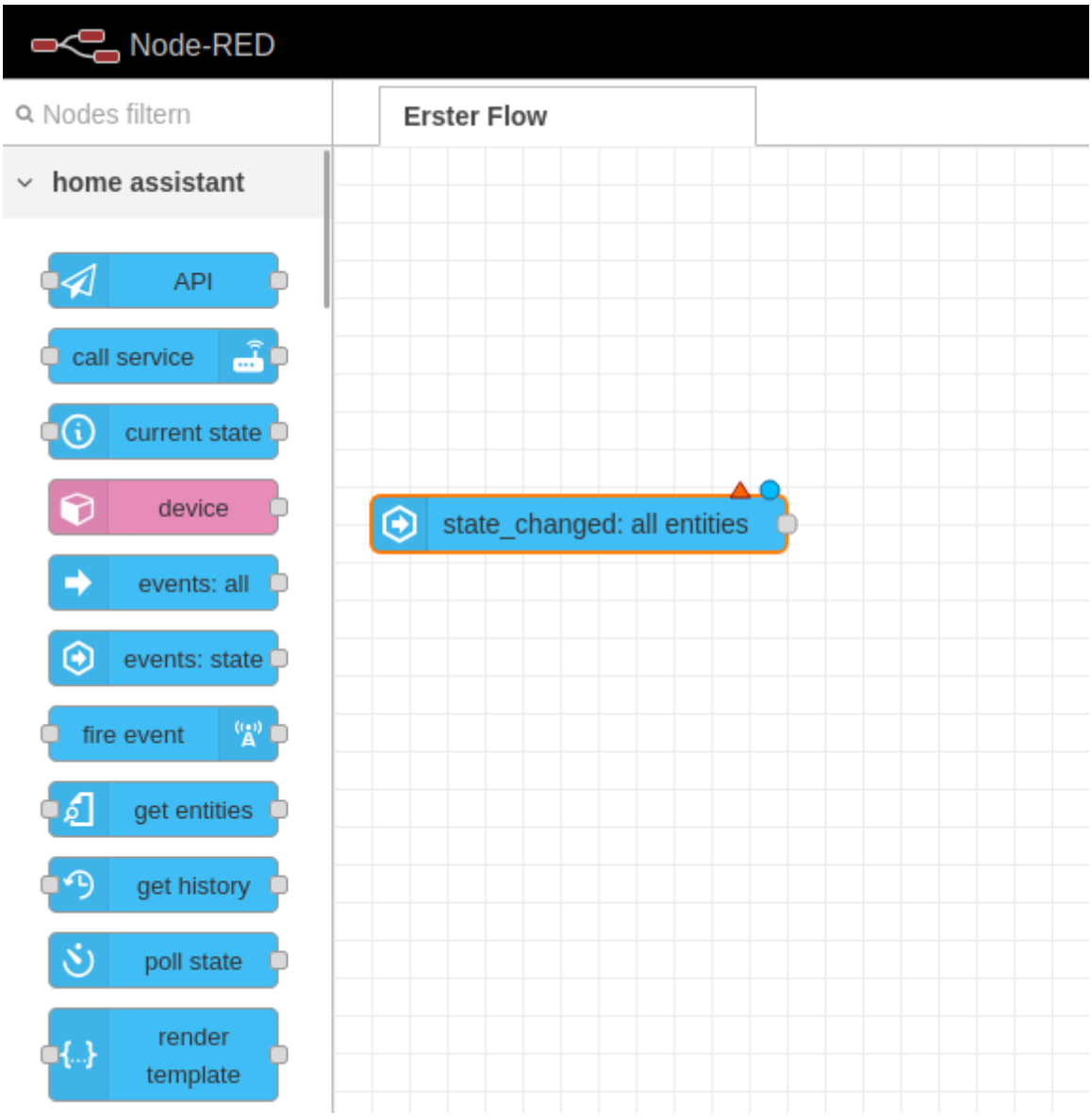
Nodes

Plazieren von Nodes

Um ein Node auf den Flow zu ziehen, den node den man haben möchte aus der Liste Links mit der linken Muastatse gedrückt halten und auf den Flow ziehen

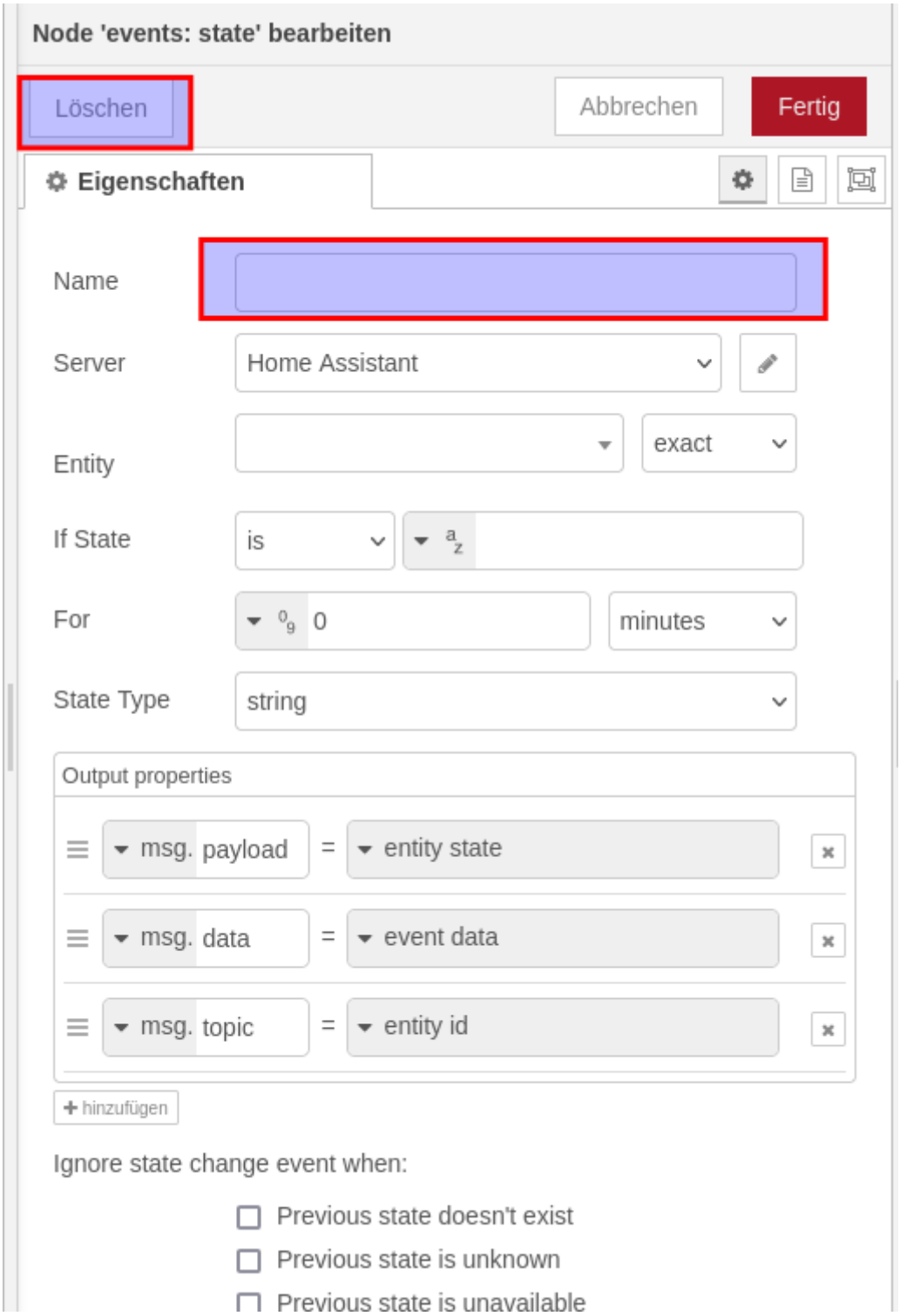


Nun ist die Node plaziert.



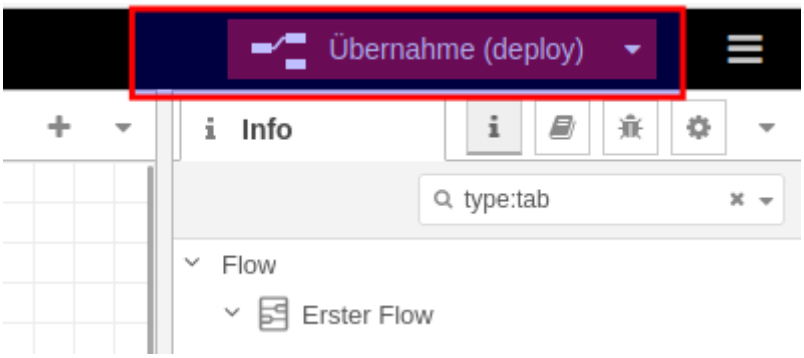
Eigenschaften der Node anzeigen, umbenennen und löschen

Hierrüber kann die Node auch gelöscht werden, analog wie beim Tab.
Dazu doppelklick auf die Node.
Nun können wir Name vergeben oder aber über den löschen Button die Node löschen.

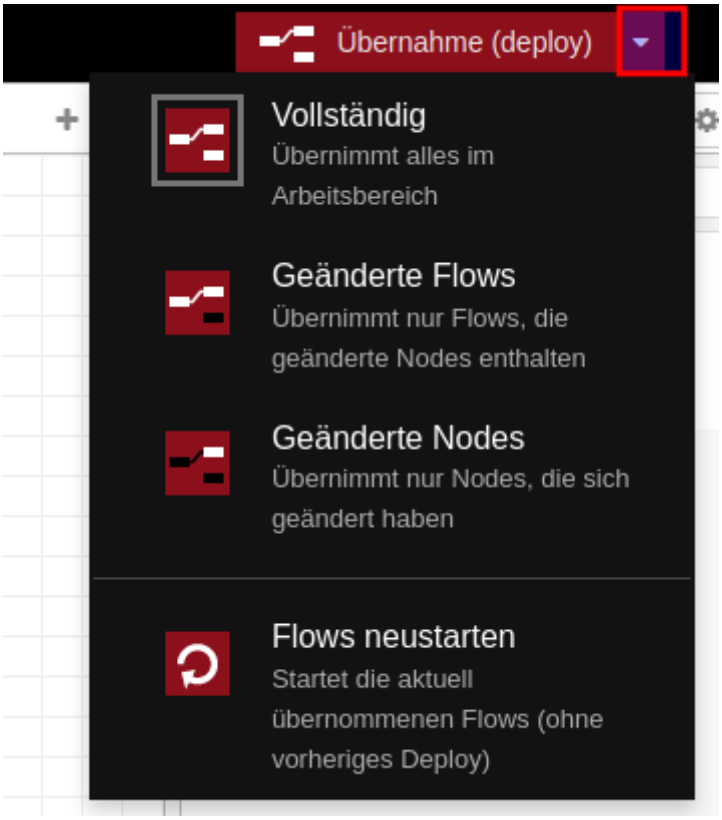


Deployen - scharf schalten

Damit Nodes eigenschften etc. auch Aktiv geschaltet werden, gibt es oben rechts den Butto deploy. Kann man gleich setzten mit speichern oder übernehmen.



Über den Pfeil kann man auch sagen, mann will nur bestimmte Bereiche Deployen möchte



Erste Automation

Denn Node RED dient ja dazu Automatisierungen Visuell zu Programmieren mit Blöcken (Nodes)
Was brauchen wir für die Automatisierung:

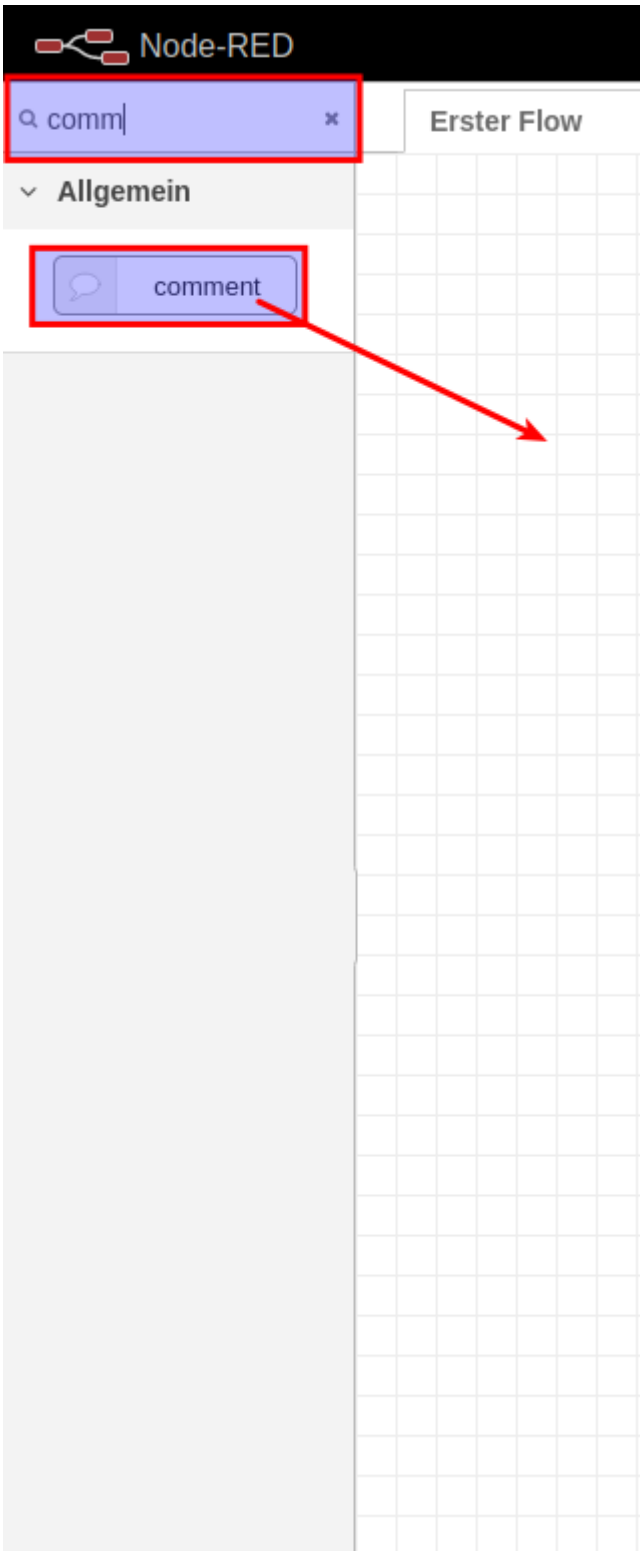
- Ein Gerät / Entität,
- Diese muss Eigenschaften haben irgendwas zu triggern also ein Sensor quasi (Zum Besipiel ein Temperatur Sensor)

- Dann wird eine Aktion ausgelöst, zum Beispiel eine Benachrichtung oder ein Gerät wird geschaltet
- Das ganze kann man dann noch mit einer Statusüberwachung prüfen.

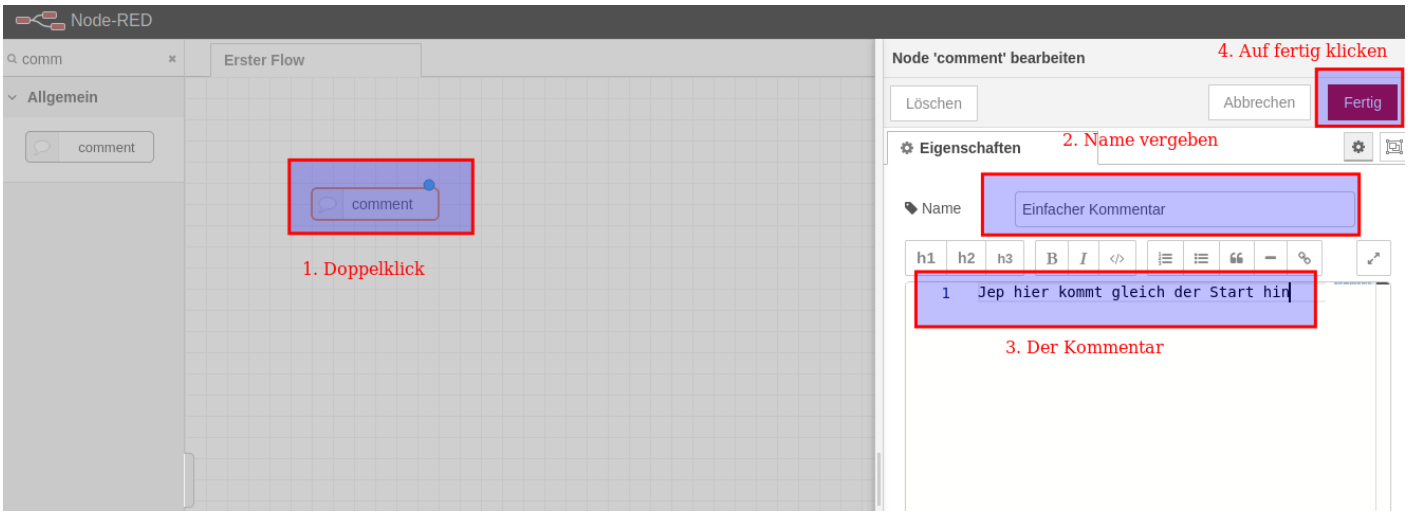
Nodes typen

Kommentar Node

Die Kommentar Node hat keine Funktion, sie dient zur Dokumentation von Bausteinen oder sonstigen infos.
Jetzt suchen wir uns die Comment Node raus. Dazu einfach comm
Oben links ist ein suchfilter. Da kann man nach Nodes filtern.
Dann braucht man nicht so viel scrollen. DIese dann wieder mit linker Taste gedrückt halten und auf die freie Fläche ziehen



Nun doppelklick auf die Node, kann man Name und den Kommentar eingeben

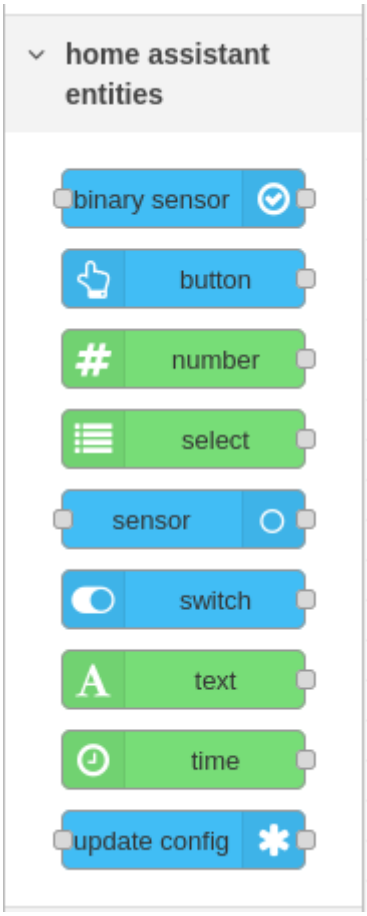


In der Node selbst wird der name angezigt, klickt man die Node an. wird im rechten Fenster der Inhalt also unser Kommentartext angezeigt.

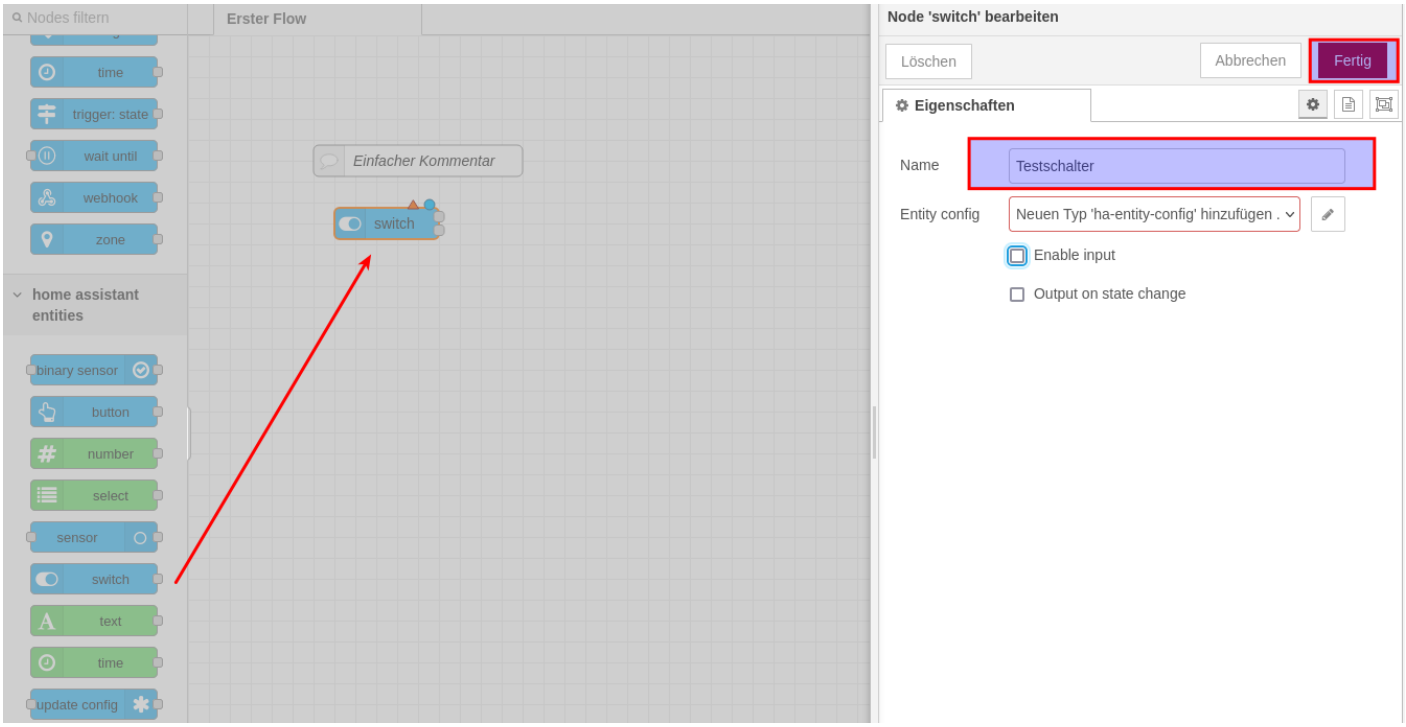
Entity Node (Ersetzt durch eine neue Gruppe von Elementen die "Home Assitant Entities")

Die Entity Node ist eine Node mit der man eine Entität in Home Assistant anlegen kann. Also ein Virtuelles Gerät.
Zum Beispiel ein Schalter oder ein sensor. Diese Entität ist allerding deprecated. Denn in diese Entität war zu unflexibel.
Dafür gibt es jetzt neue Nodes, mit denen Man GUI Elemnte oder Sensororen in Home Assistant bauen kann.
Virtuell halt.

Am Namen kann man schon erkennen was was sein soll.

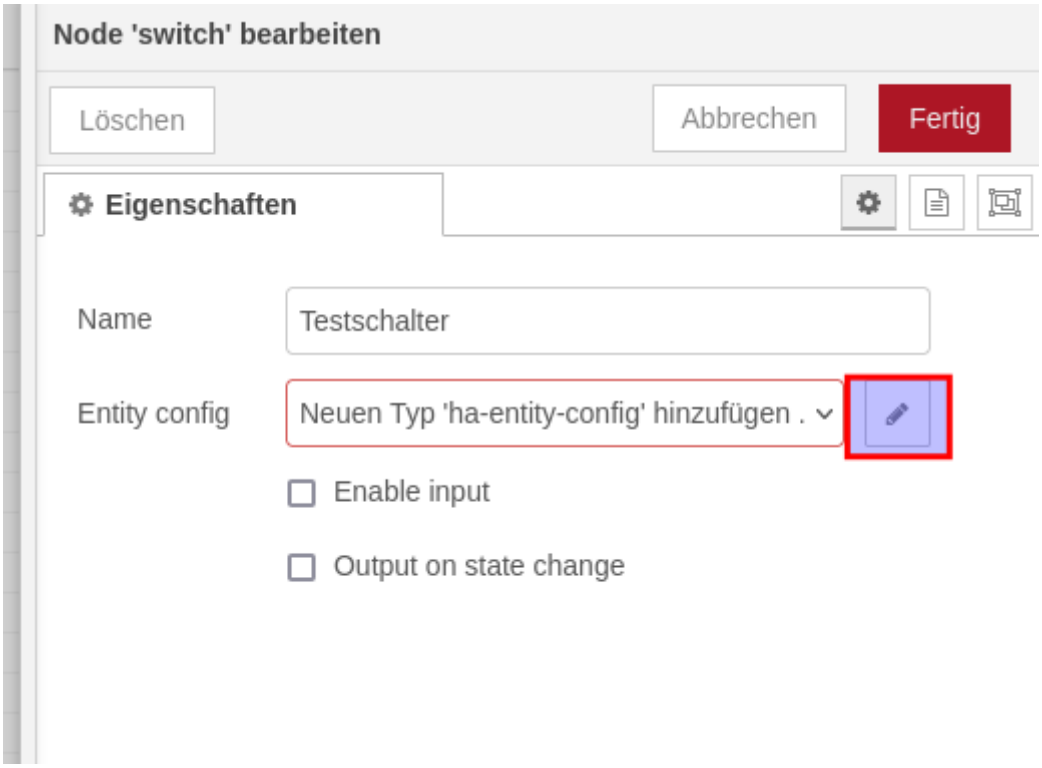


Wir ziehen uns einen Schalter (switch) auf die freie Fläche) doppelklcik drauf und geben Ihm den Namen. Testschalter.

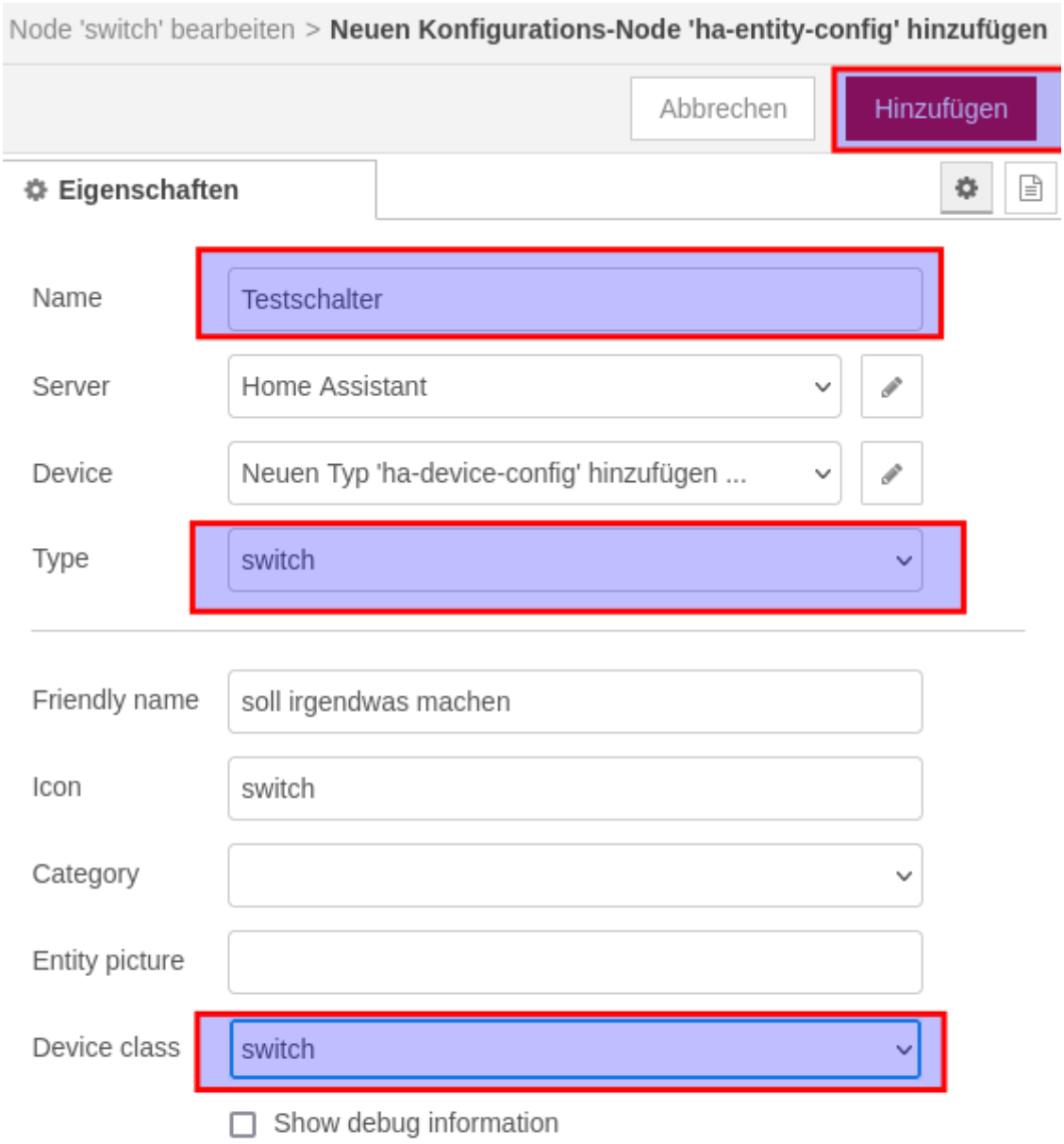


klicken dann auf den Bleistift.

In Neueren Versionen ist dies ein + Button

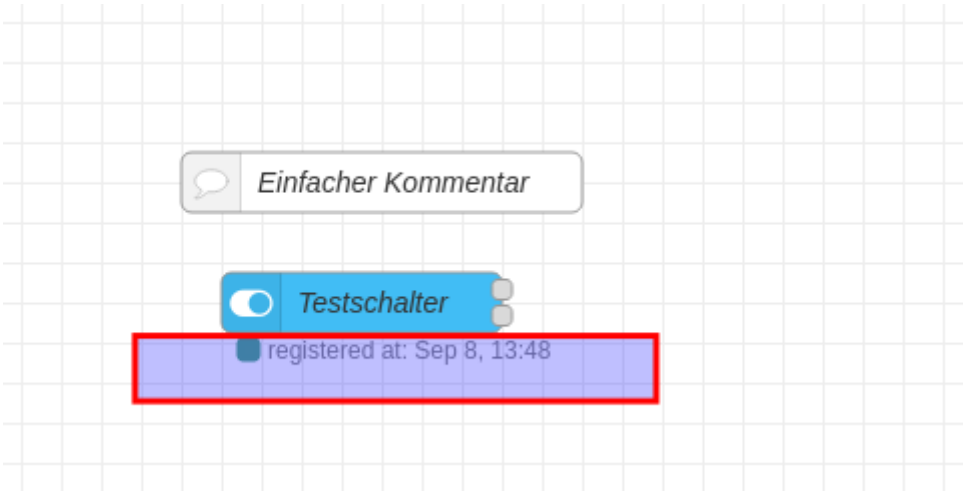


Nun Einen Namen angeben, switch auswählen und Device class auch switch.
Mit dem Stift bei Device kann man auch einen Eigenen Geräte Namen und Hersteller angeben.
Hier ist es jetzt in neueren Version auch ein + Button anstatt Stift
Wenn man möchte. Dann auf Hinzufügen

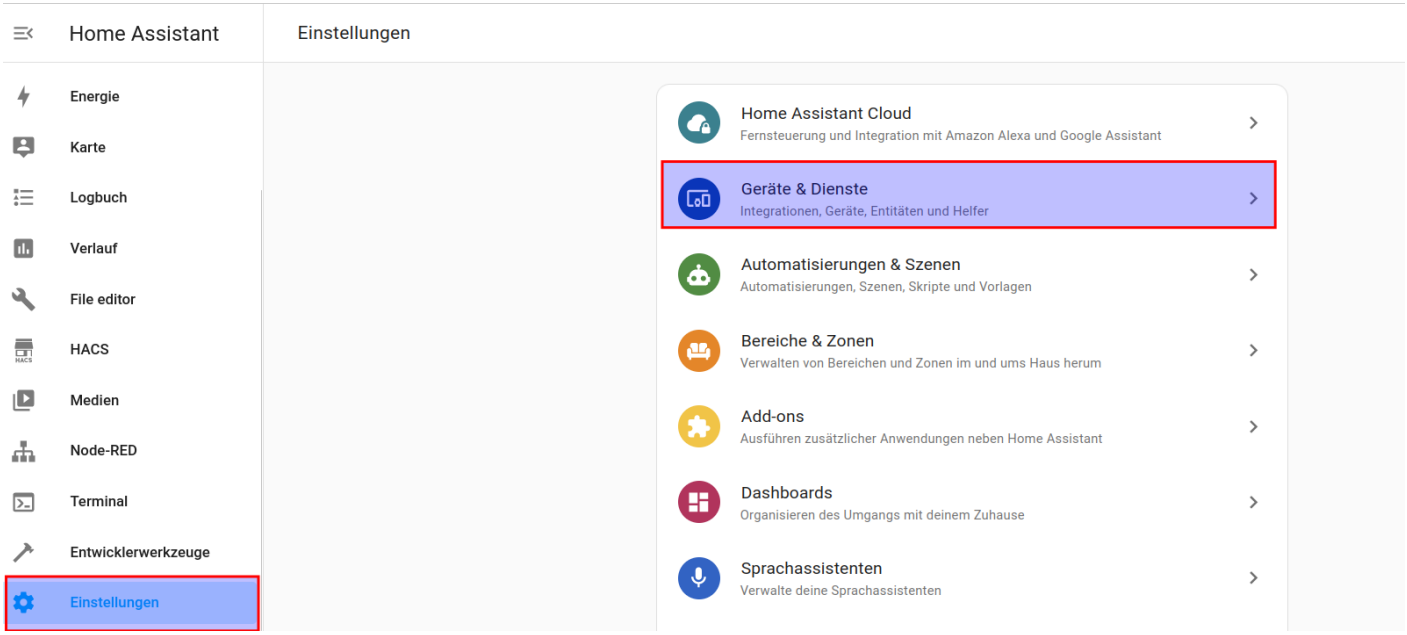


Nun noch Auf Fertig klicken.
Danach auf deploy klicken

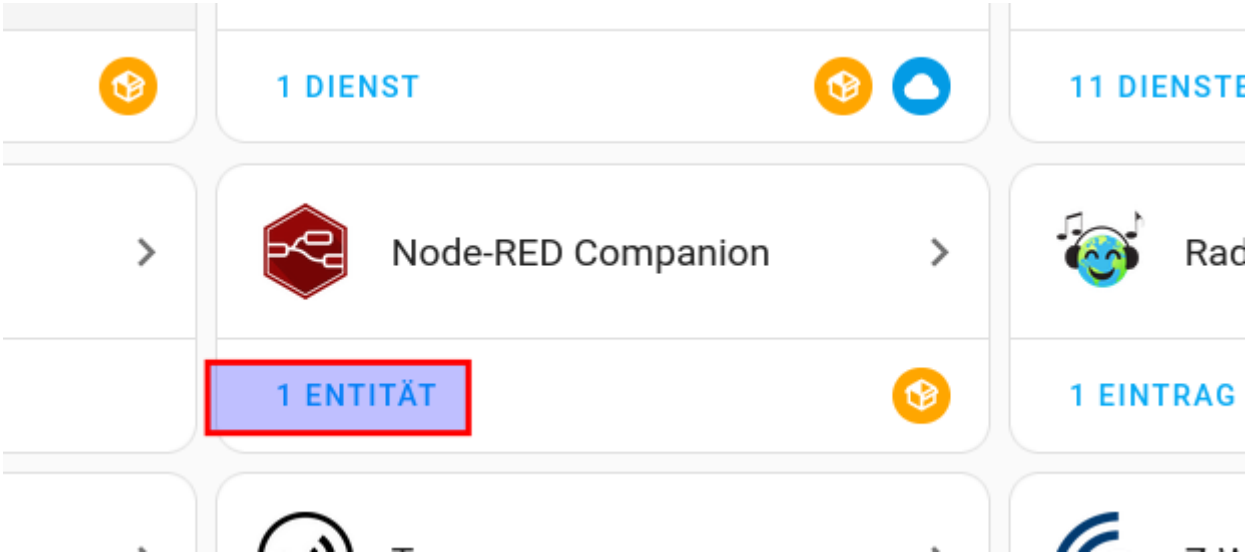
Nun haben wir einen registrierten Schalter



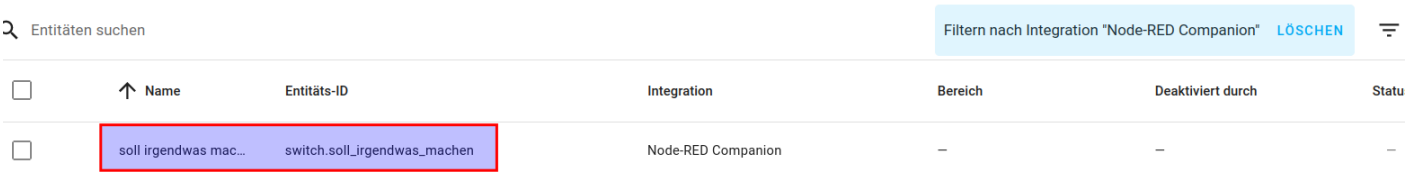
Um zu sehen welche Entitäten von Node-RED Companion es gibt, gehen wir unter EInstellungen -> Geräte und Dienste



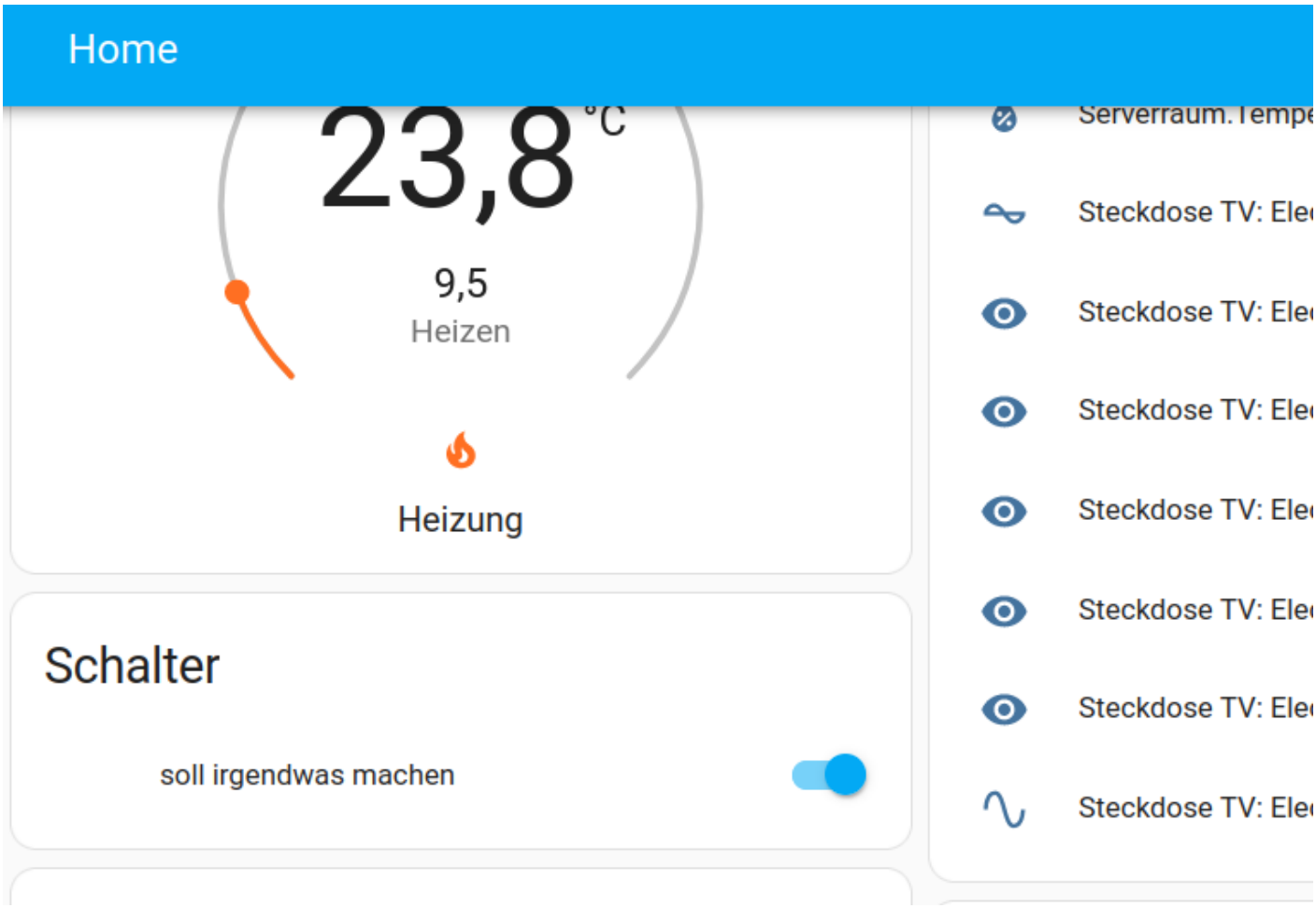
Dort sehen wir dann bei Node Red Companion 1 Entität. Weil wir haben zur Zeit einen Schalter. Da klicken wir drauf.



Nun sehen wir unseren Schalter. Name soll irgendwas machen. vom Typ switch.{name} name = soll irgendwas machen.
läuft. Dieses Gerät kann wie jedes anderes Gerät dann auch ein Bereich zugewiesen werden usw.



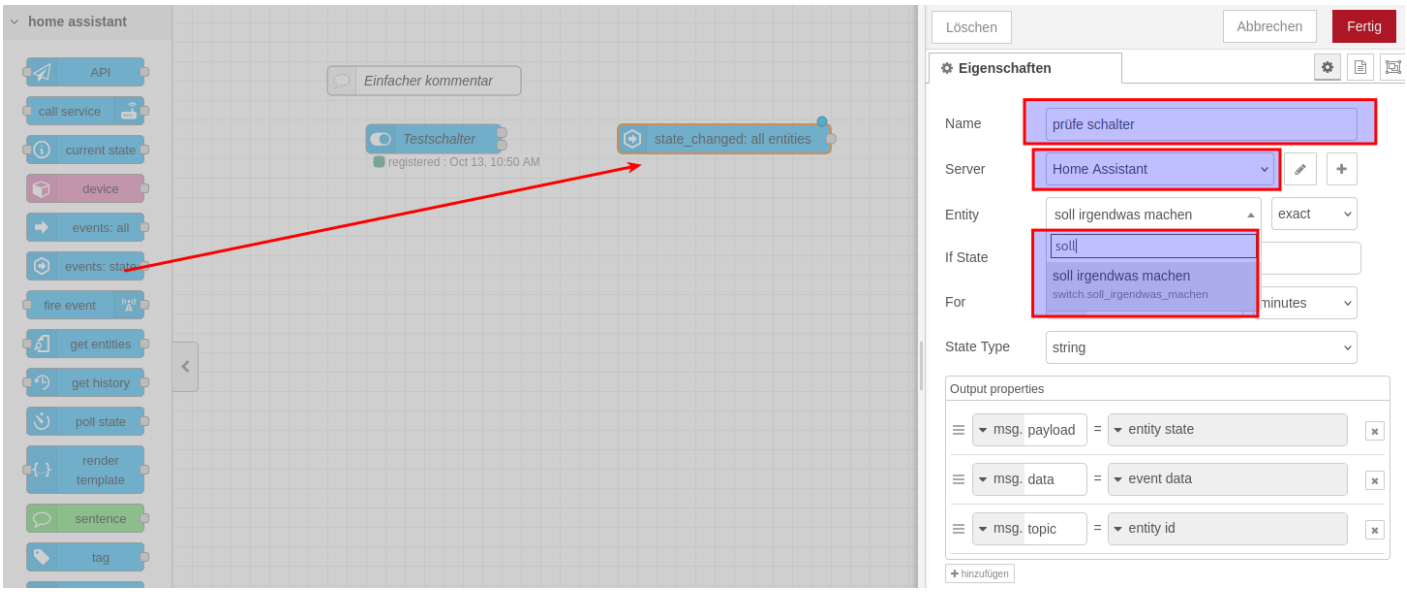
Auf dem Dashboard sieht das dann so aus. Ein Virtueller Schalter.



Event State Node (trigger/sensor) Teil 1

Die Event State Node fragt Geräte eigenschaften / Zustände ab, also ein trigger.

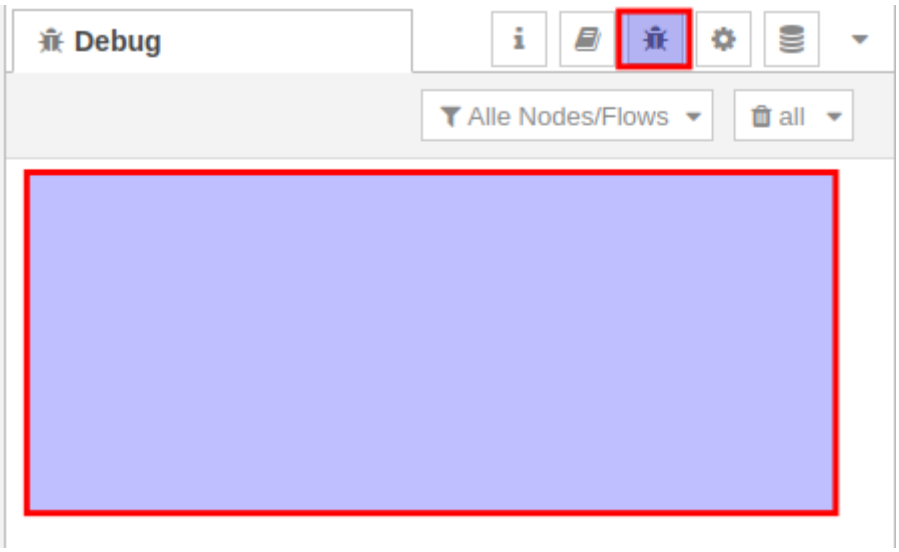
Wieder eine Node hinzufügen, hier die Event State Node einen Namen vergeben, Server auswählen (HomeAssistant)
und das Entity auswählen.
Über die suche hab ich jetzt einfach "soll" eingegeben. Und schon bekommen wir unseren Schalter soll irgendwas machen.
Wir klicken erstmal auf fertig.



Debug Node

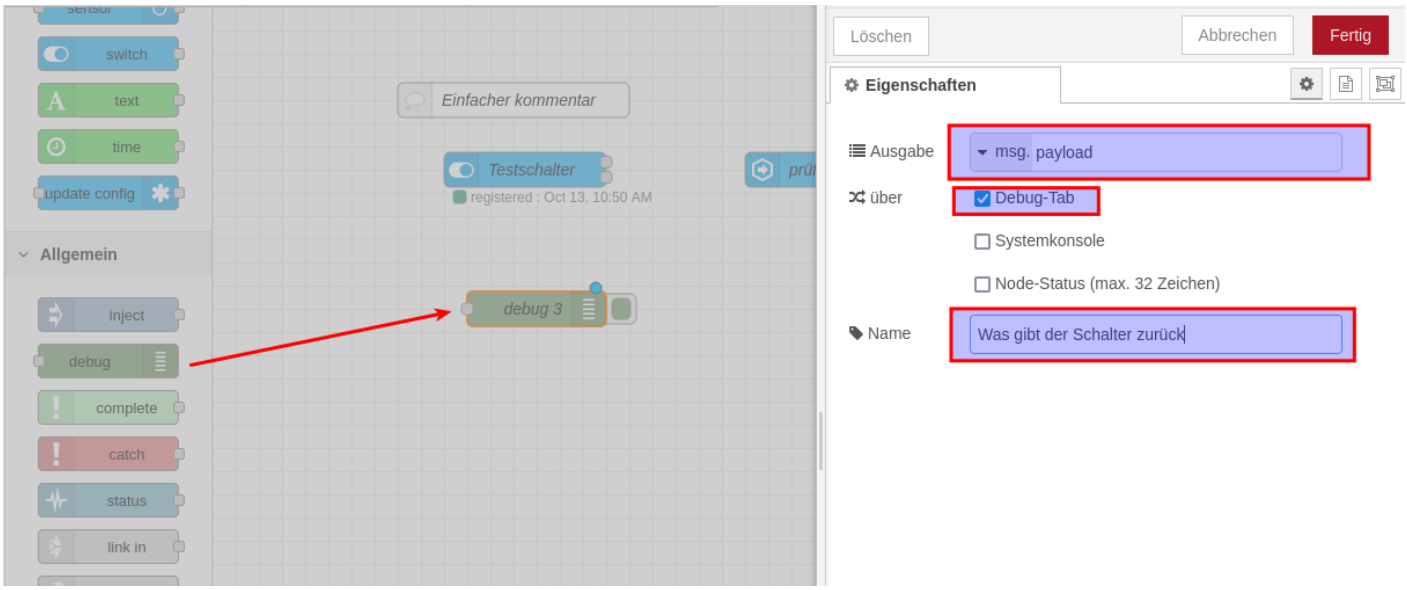
Bevor es an die Bedingung geht.
Was für einen Wert gibt unser Schalter denn zurück?
Das bekommen wir mit der Debug Node raus.
Diese Node ist besonders hilfreich um zustände zu bekommen.

In der Seitenleiste Rechts gibt es Symbol mit nem Käfer, da landen alle Infos von Debug Nodes oder Fehler von Nodes.
Über die Mülltonne mit all, lassen sich alle Nachrichten löschen, sprich das Debug Log leeren.

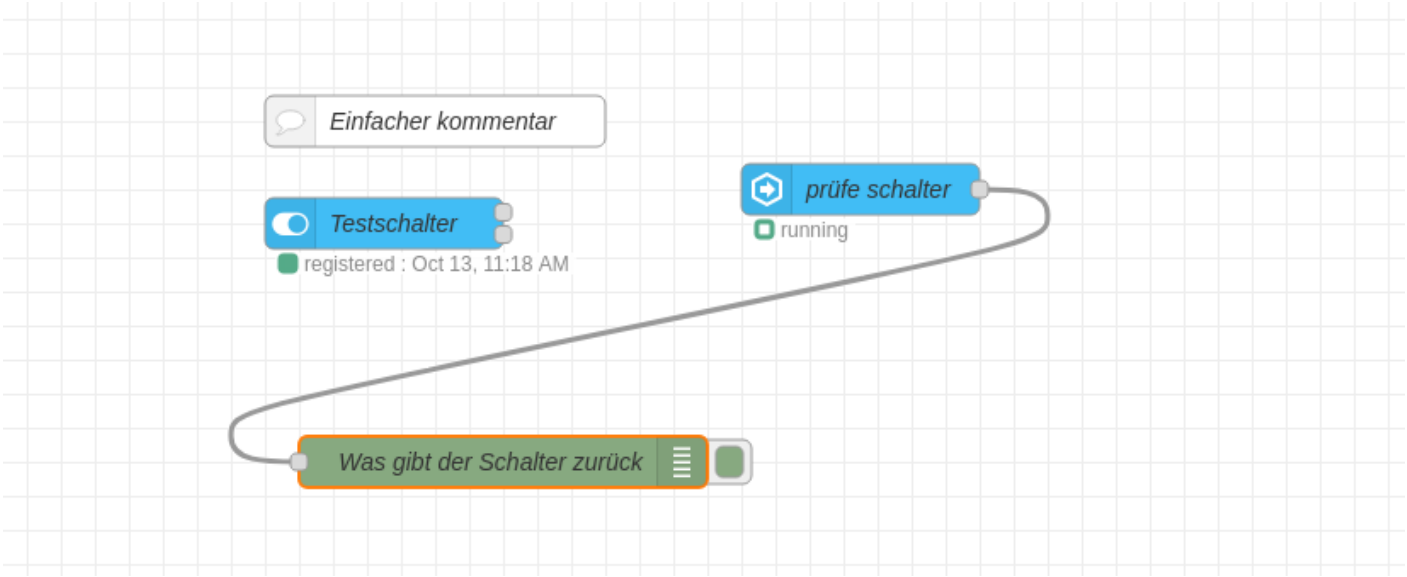


Für Fehler braucht man keine Debug Nodes.

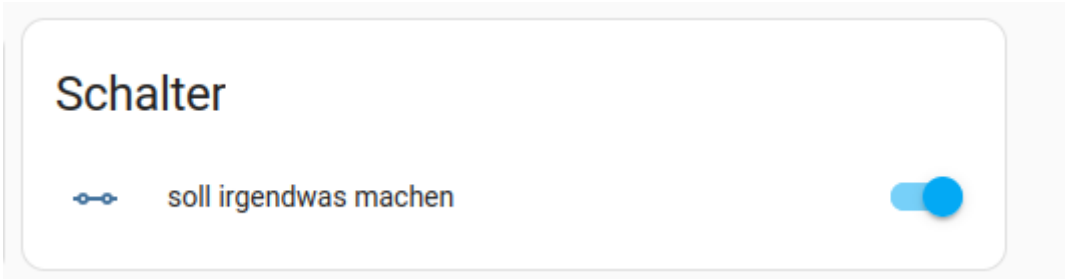
Wir fügen eine Debug Node hinzu und geben Ihr einen Namen.
Als Ausgabe wollen wir meisten den payload, also sprich die Daten die das Gerät zurück gibt.
Den Haken bei Debug Tab rein, so das diese Dann auch im Debug Log hinzugefügt werden.
ist der Haken raus, steht der Zustand nur unten drin



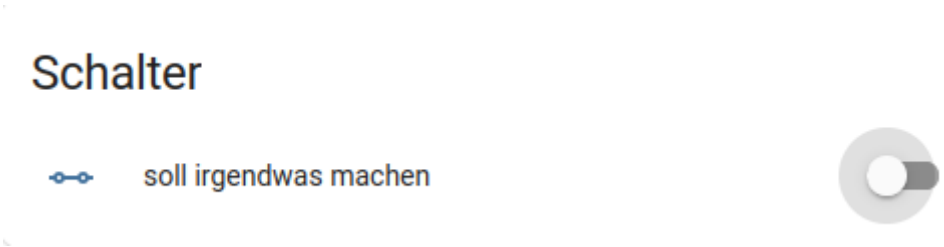
Nun den Ausgang unserer prüfe schalter node mit dem Eingang des Debug Nodes verbinden und auf Deploy klicken



Nun einen neuen Tab öffnen und unseren Schalter aus und wieder einschalten betätigen.
Nach Erstellung ist der Schalter Standardmäßig eingeschaltet



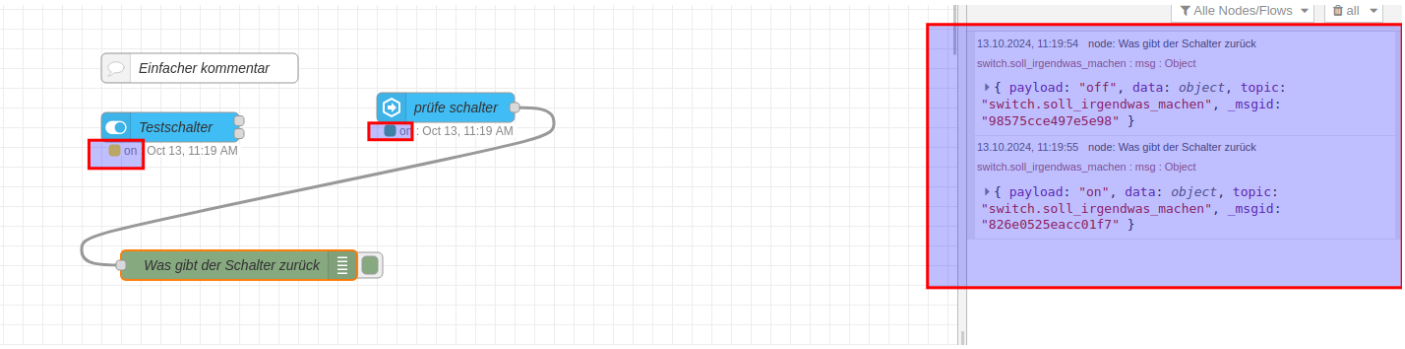
Nun machen wir den Schalter aus



Und wieder ein

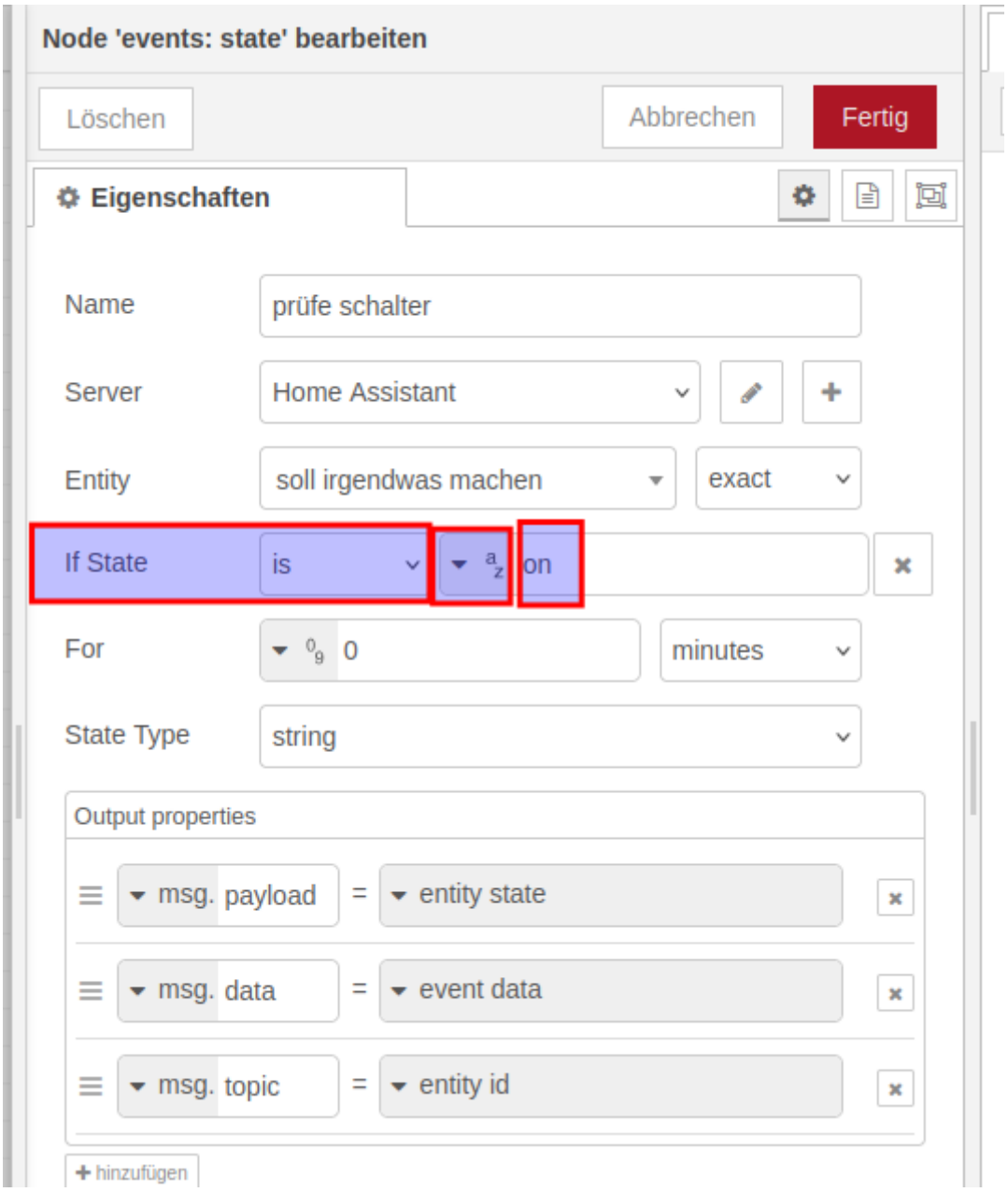
Im Debug ist nun folgendes zu sehen.

Für einfache Zustände braucht man tatsächlich keine Debug Node, aber es geht ums Prinzip, denn wie zu sehen, wir der Wert unterm Testschalter angegeben, allerdings immer nur der letzte.
Übers Debug Node wissen wir nun beide zustände. Wenn der Schalter eingeschaltet ist ist es der Wert on und wenn er ausgeschaltet ist , ist es off.
Es hätte ja genauso gut true and false sein können.
Da der Wert on off ein Text ist, wissen wir das wir einen String vergleichen müssen.



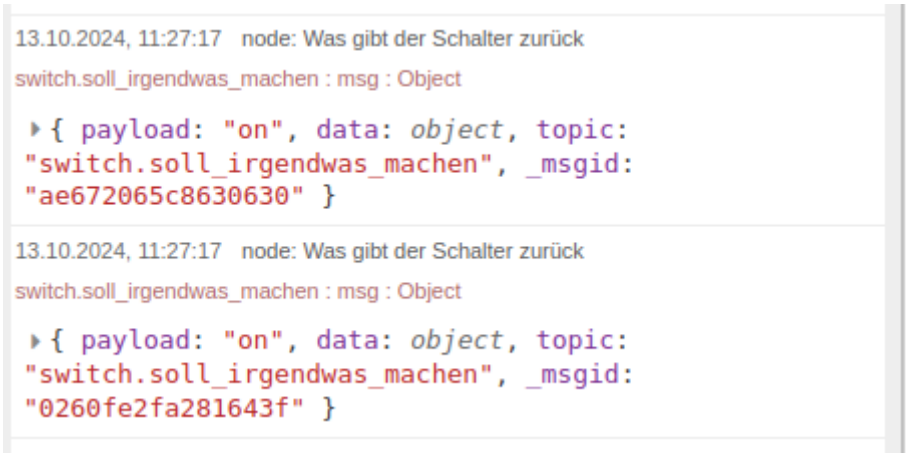
Event State Node (trigger/sensor) Teil 2

Nun können wir wieder unseren Service anklicken und die Bedingung eintragen. Ist Text und on.

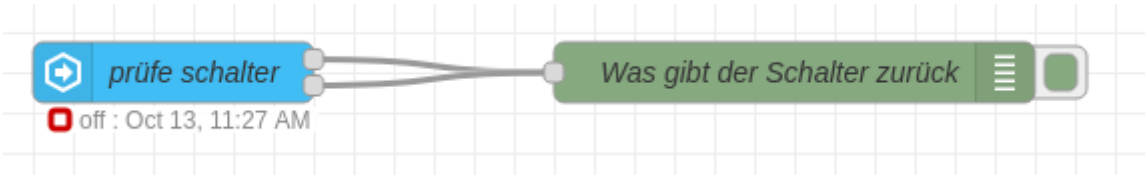


Unsere Node hat zwei Ausgänge.
Der Erste wenn die Bedingung wahr ist und die zweite Unwahr.
So können wir zwei Zustände erfassen. Ist der Schalter an, mach das, ist der aus mach das.

Nachdem wir die Bedingung eingetragen haben, bekommen wir im Debug auch nur noch on.
Denn vorher war die Node ja immer true und der Payload wurde für on wie auch off an dem ersten Ausgang übertragen. Wollen wir im Debug Node auch jetzt noch beide zustände müssen wir den False Ausgang auch noch ans Debug Modul packen.

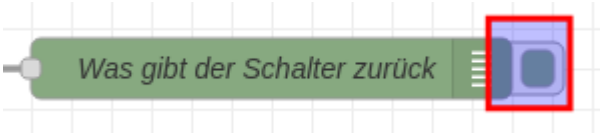


Beide Ausgänge aufs Debug Modul.

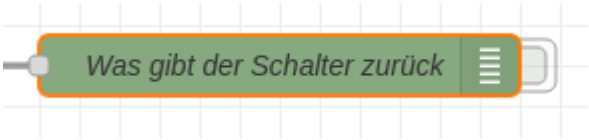


Jetzt wäre ein Kommenta sinnvoll und das Debug modul kann gelöscht werden, wenn man es nicht braucht, oder deaktiviert, damit das Log nicht vollgemüllt wird.

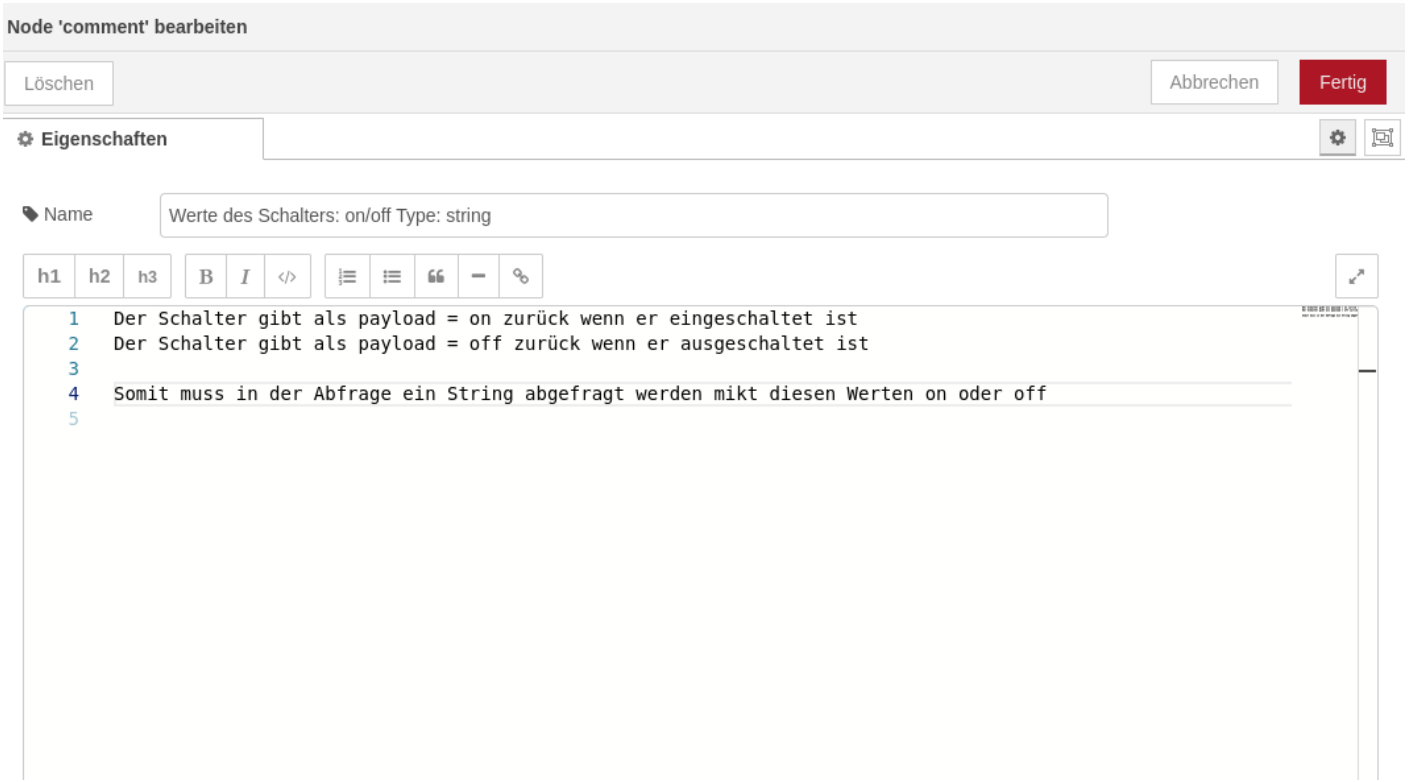
Deaktivieren



Nun ist es deaktiviert. Deployen nicht vergessen.



Ich lösche das Debug Objekt aber und ändere unser Kommentarfeld was jetzt auch einen sinn macht

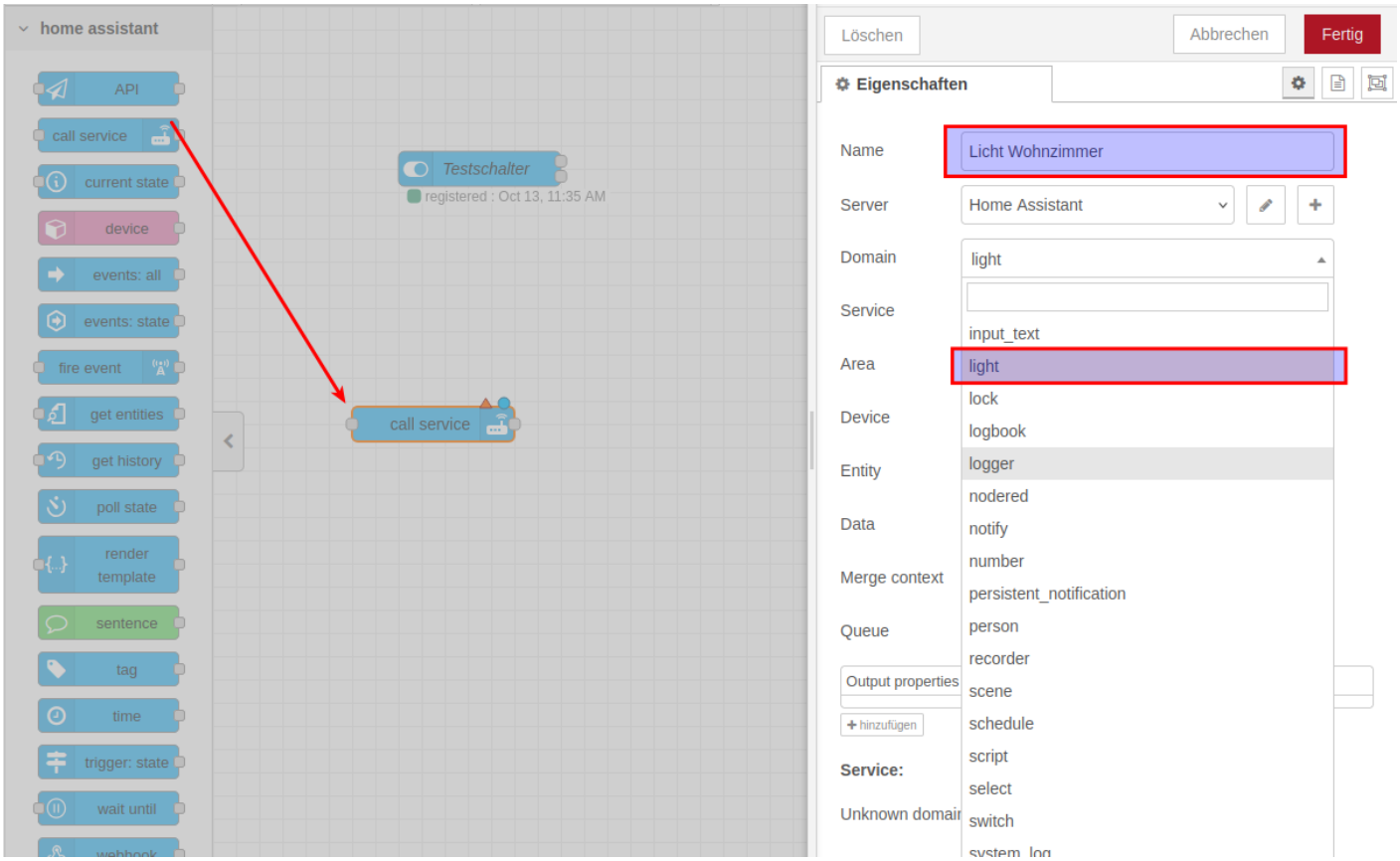


Ich schreibe oben in den Titel schon die Werte rein. Für die Vollbeschreibung dann doppelklick auf das Kommentar Node

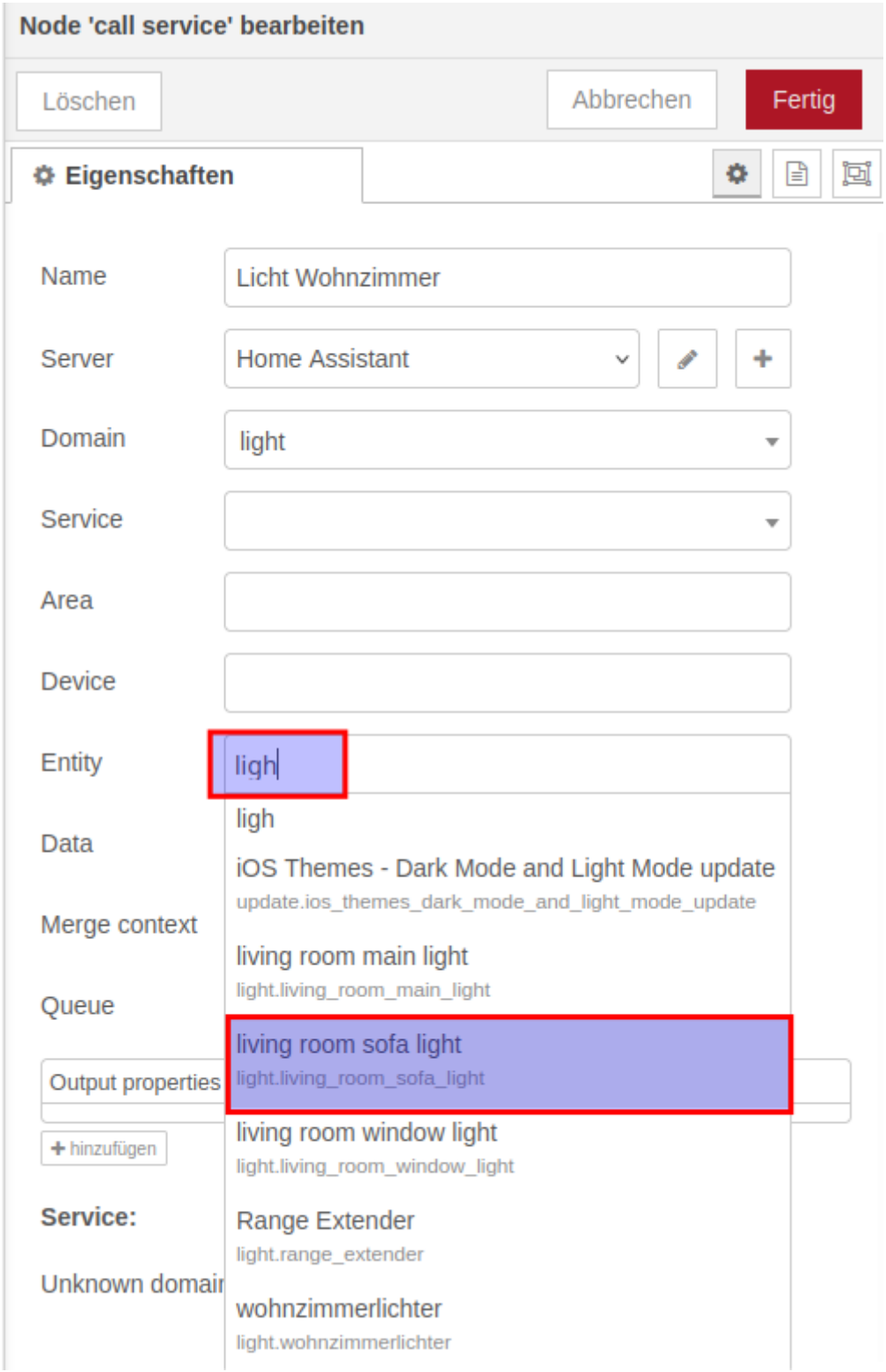
Nun können wir endlich unsere Aktion hinzufügen.
Das machen wir mit der call service node

Call Service Node

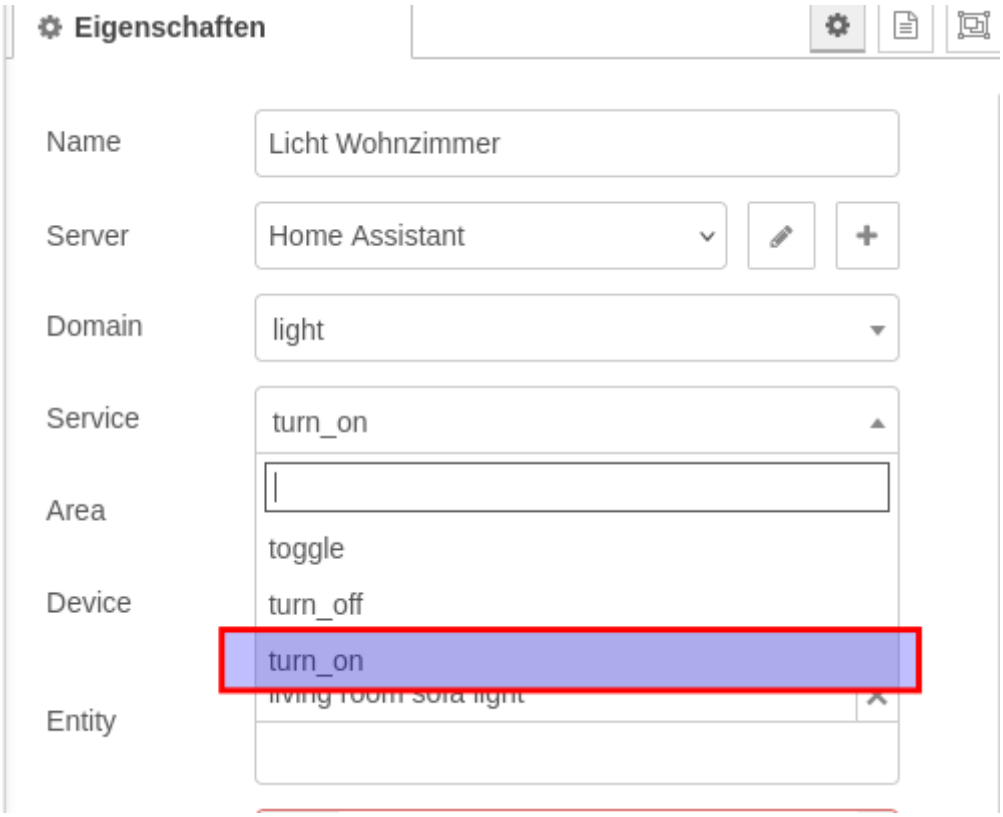
Wir fügen ein call service node hinzu vergeben einen Namen und die wählen unsere Geräte Klasse, Hier Licht



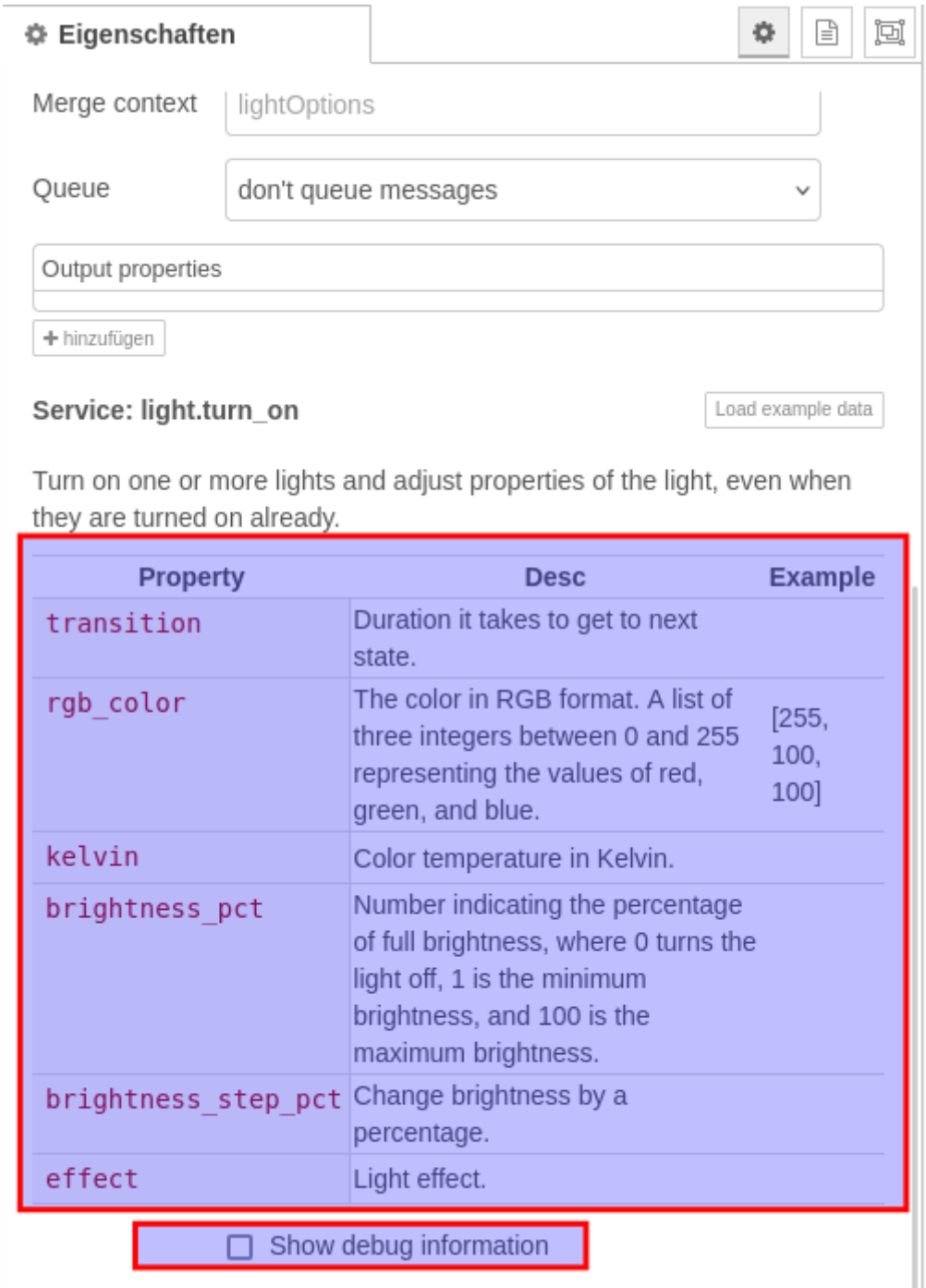
Bei Enitity wählen wir das eigentliche Gerät.
ich gebe in die suche light an, so werden mir alle Geräte mit light aufgelistet.
ich wähle mein Sofa licht. Auch Gruppen können ausgewählt werden wie meine Gruppe light.wohnzimmerlichter



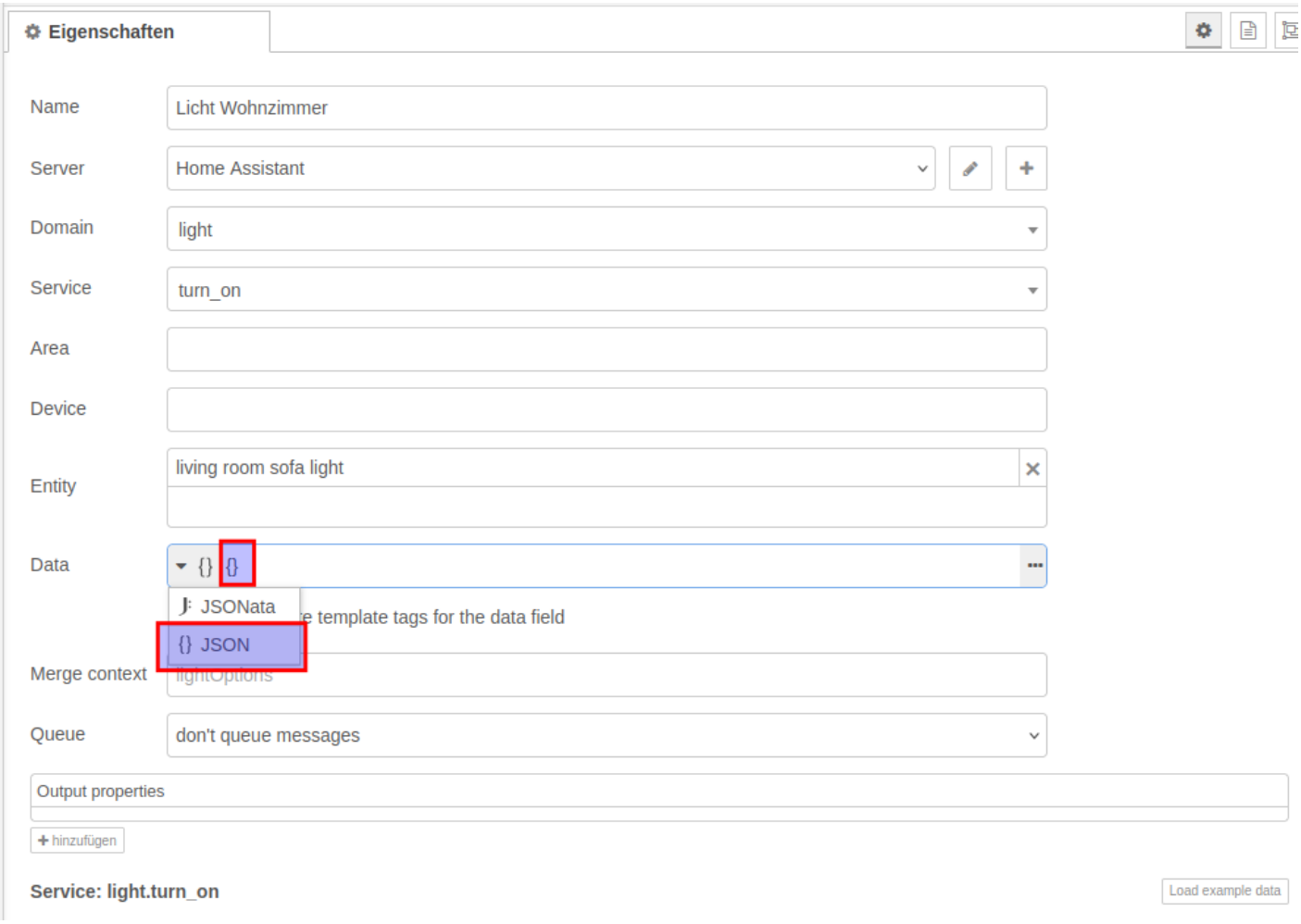
Nun wählen wir den Service aus. Da wir gesagt bei status on soll das licht an gehen und nicht togglen, wählen wir hier on



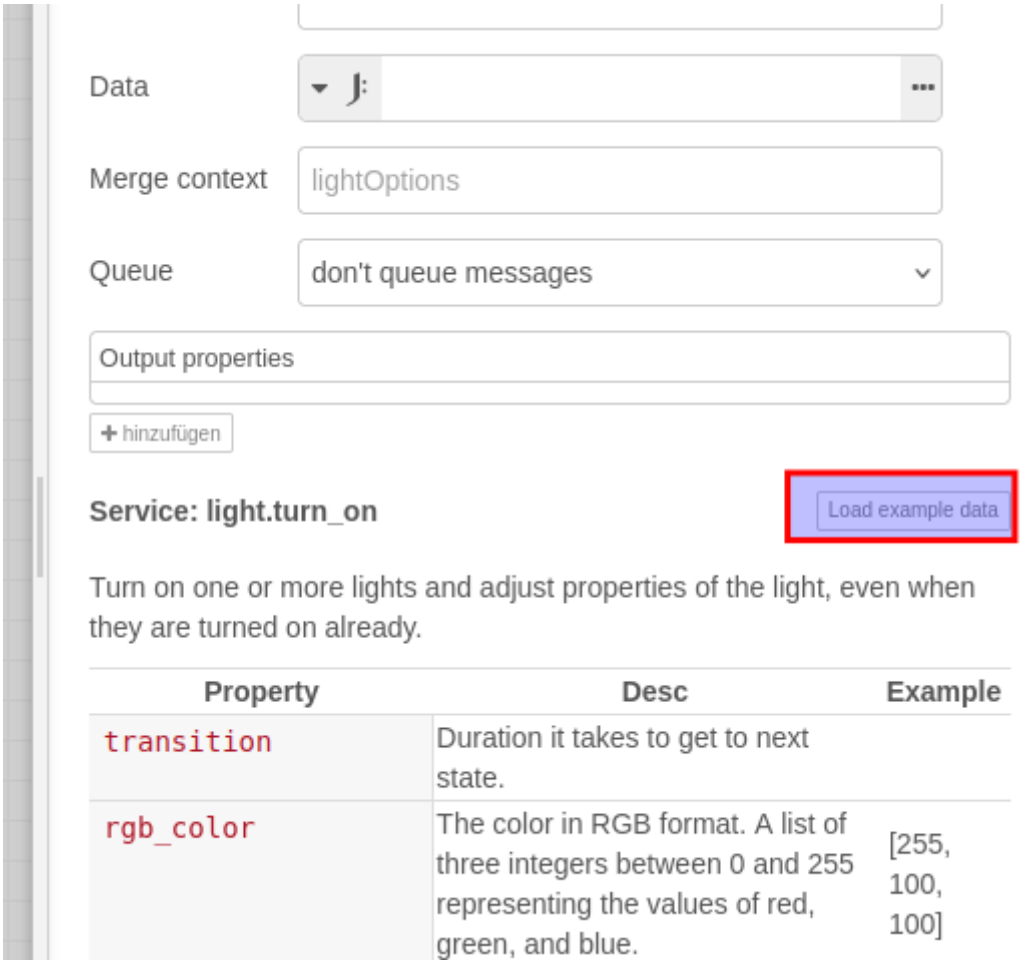
Nun da wir den Service ausgwählt haben, haben wir weiter unten bei Data eine reihe von Werten die Wir übergeben können angezeigt.
Die Service node hat übrigends auch die Möglichkeit Debug Informationen anzuzeigen. So braucht man keine Extra Debug Node erstellen.
Viele Nodes haben das schon inbegriffen.
Hier aber eine Liste der Werte, die dieser Service unterstützt, einige Werte werden eventuell von dem Entity nicht unterstützt.
Dann passiert halt nichts und wird nur eingeschaltet.
zum Beispiel bei einer weißen Lampe die keine Farben unterstützt, kann ich auch keine Farbe setzten:



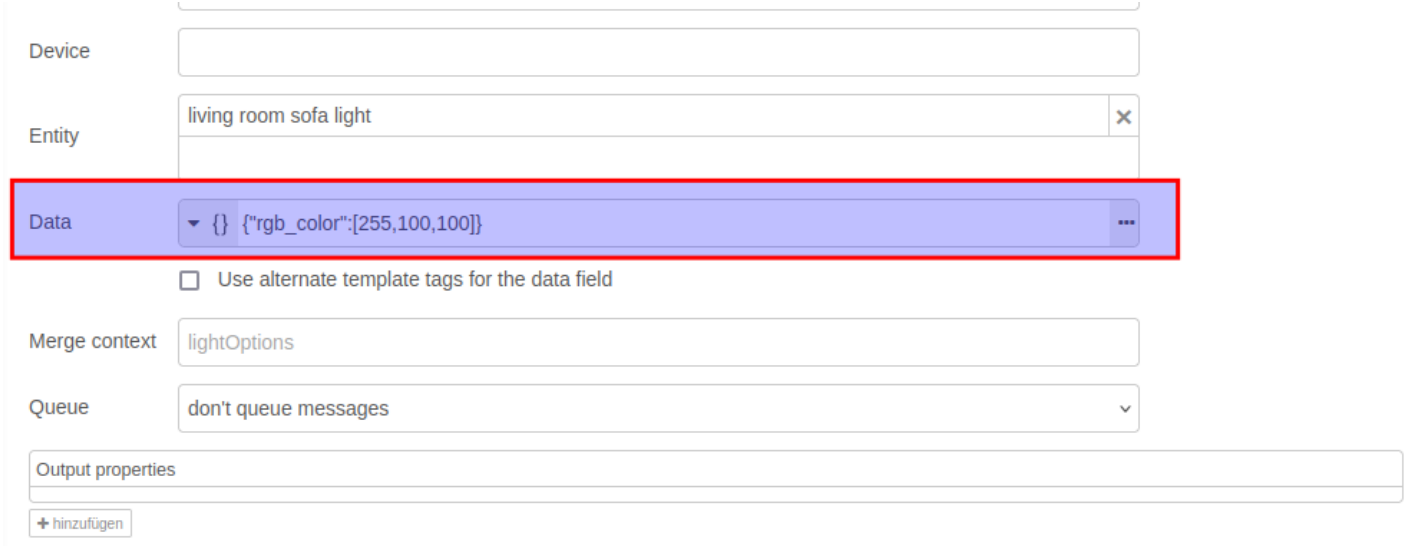
Hier sind zufällig alle Parameter optional, das heißt in das Feld Data einfach zwei geschweifte Klammern.
Wichtig: Bei Data Format muss Json ausgewählt sein, nicht json Data



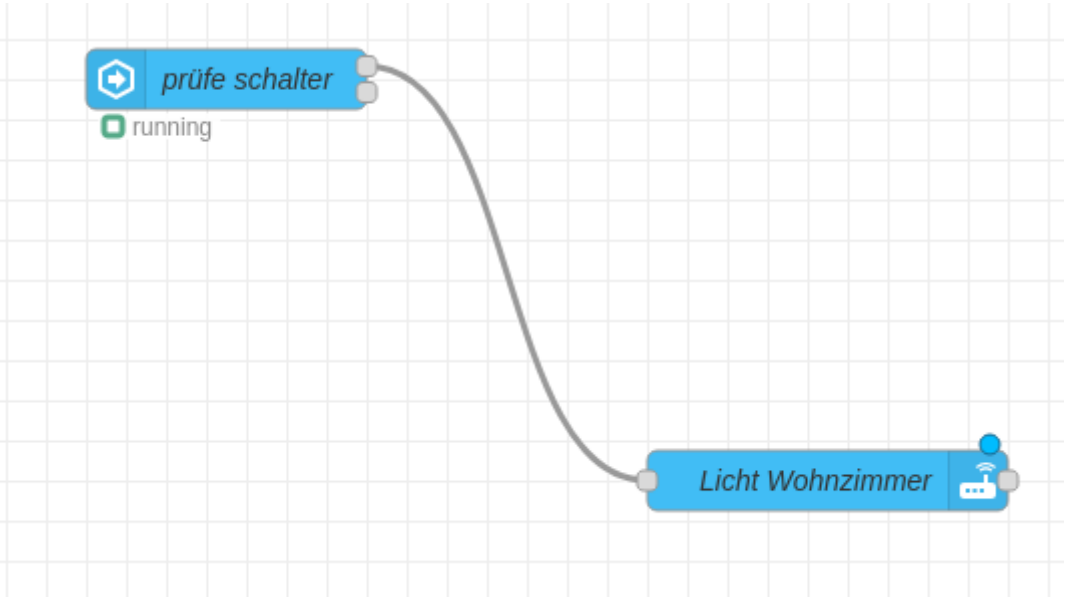
aber möchte man ein Beispiel, einfach auf Load Example Data klicken.
Das find ich total Praktisch, dann braucht man nur noch die Werte ändern.



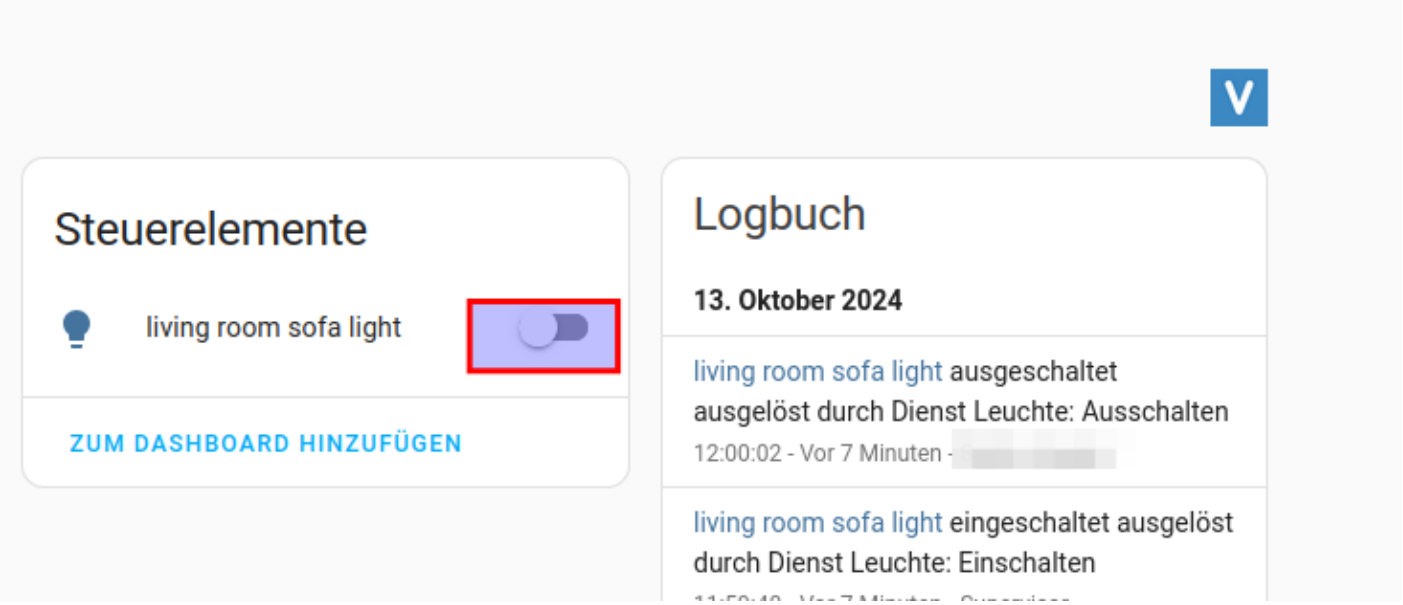
Nun braucht man nur noch den Farbwert ändern als Beispiel



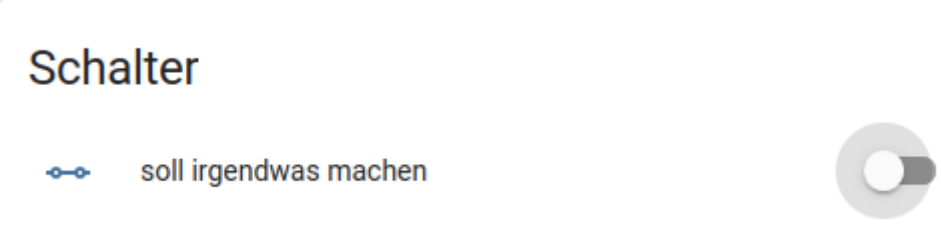
Wir belassen es bei leeren klammern.
Nun auf Fertig.
Jetzt können wir event state Ausgang wahr mit dem Licht verbinden und deployen das ganze.

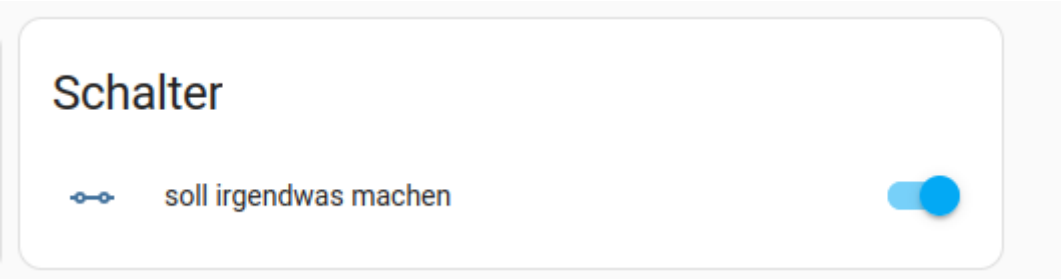


Jetzt machen wir zwei Tabs auf. Einmal wieder zu unserem Schalter und dann den Gerätestaus vom Licht.
Wir schalten unser licht als erstes aus, auf der Geräte Seite

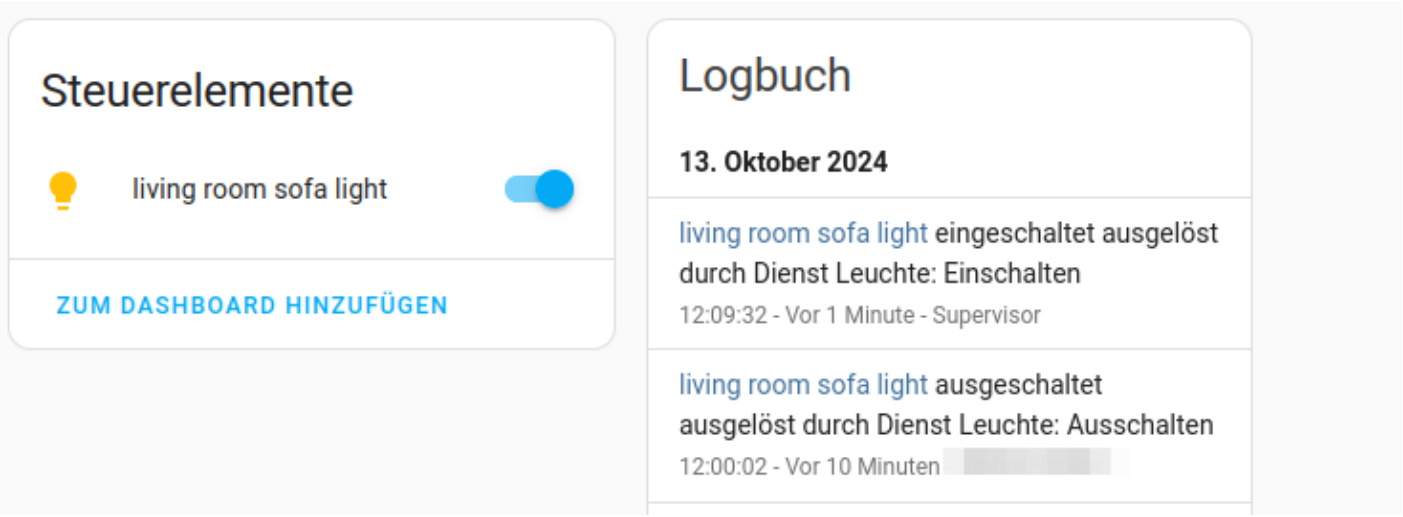


Nun gehen wir aufs Dashbaord und schalten den schalter aus und wieder ein.

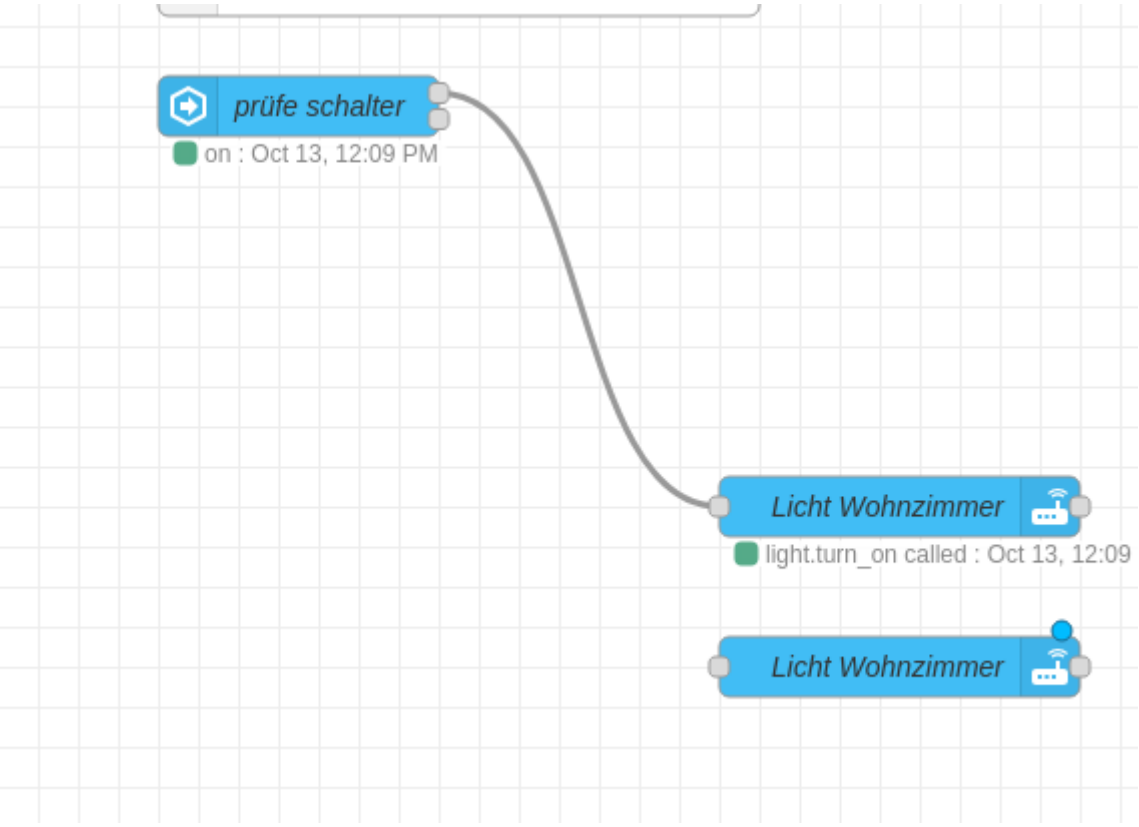




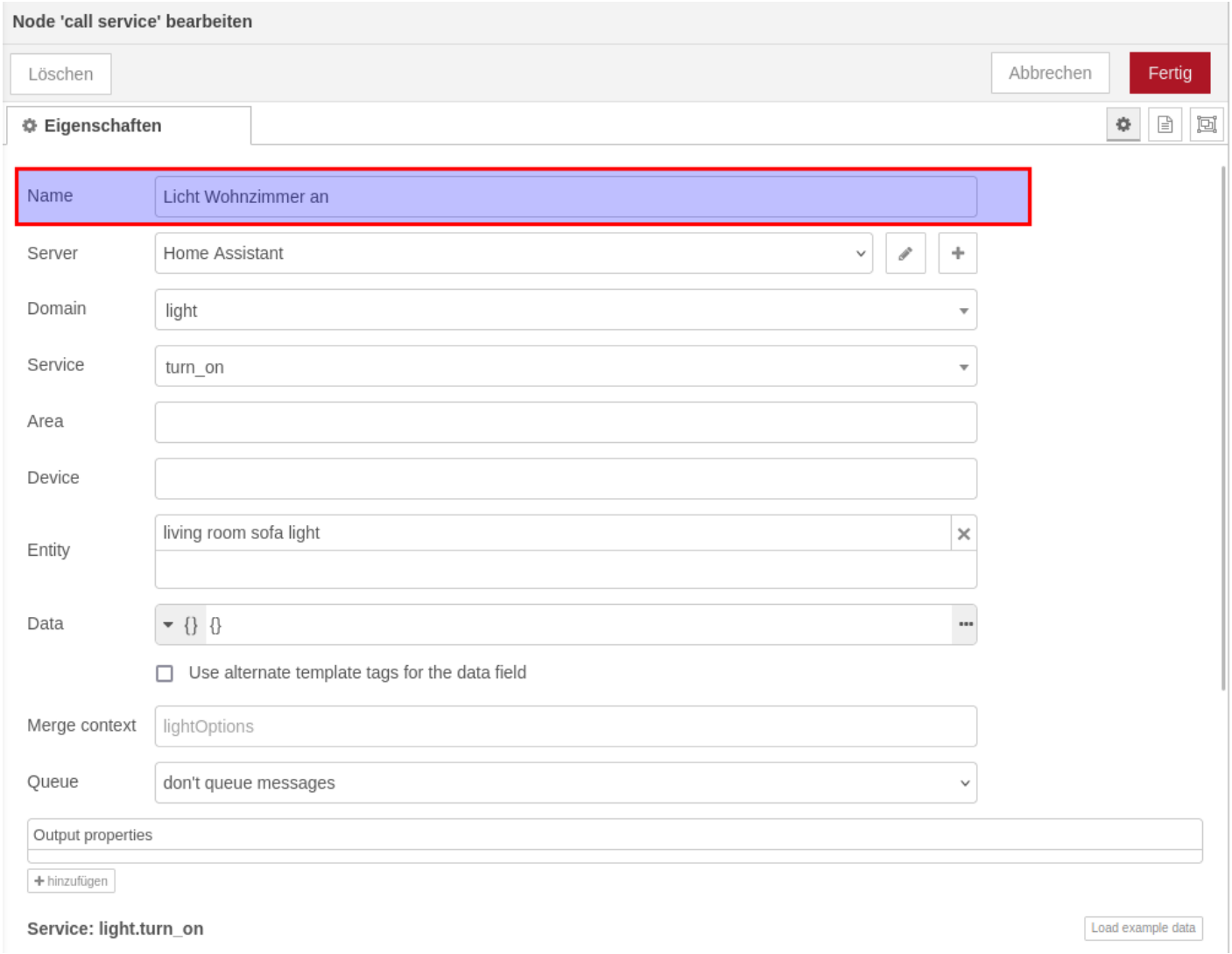
Nun wurde auch die Lampe eingeschaltet, siehe Geräte Seite



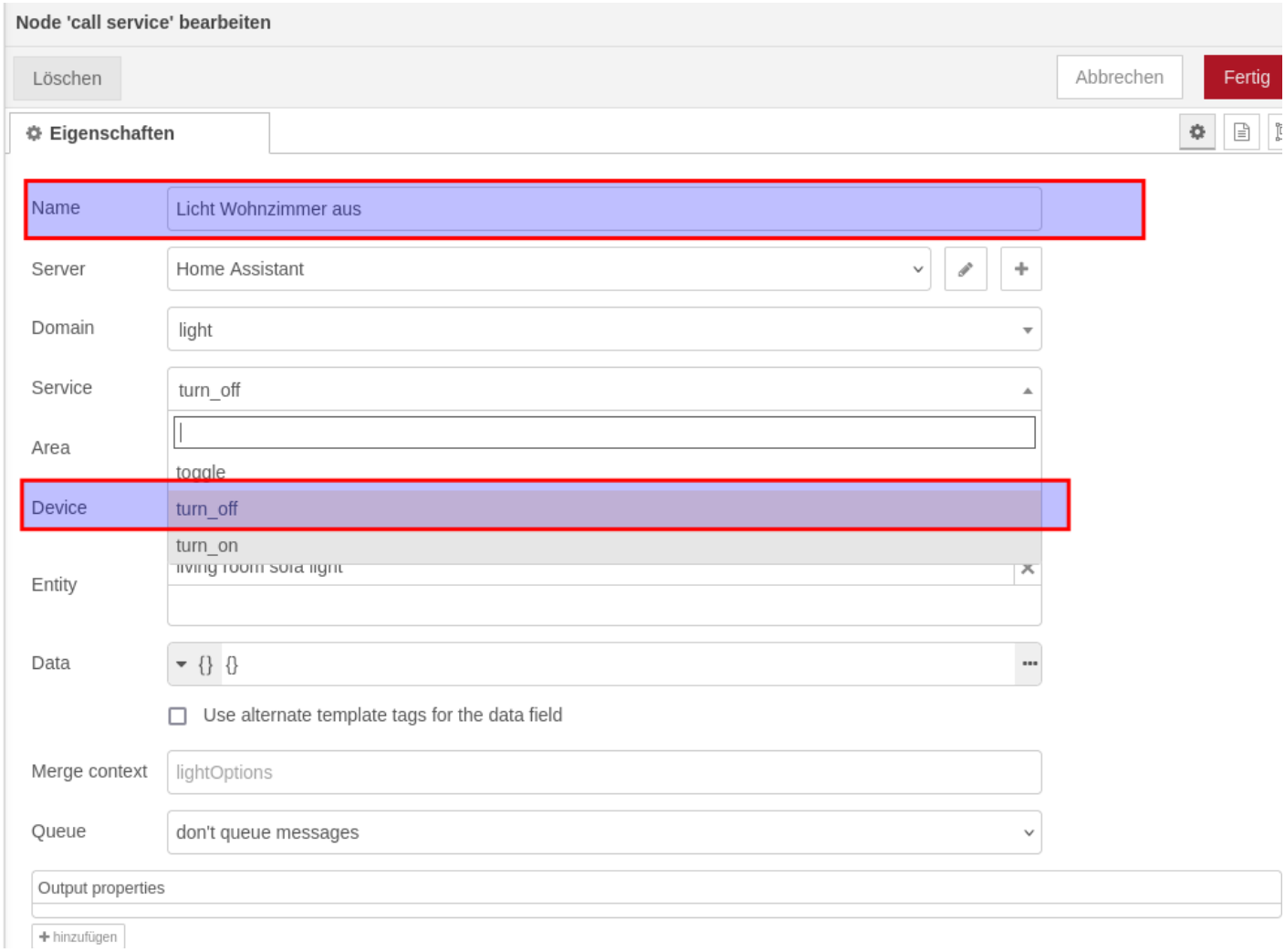
Nun möchten wir natürlich das beim ausschalten des Schalters die Lampe auch ausgeht.
Dazu kopieren wir die Node Licht Wohnzimmer. Anklicken strg+c bzw bei mac command+c und strg+v bzw command+v
Jetzt haben wir die Node zweimal



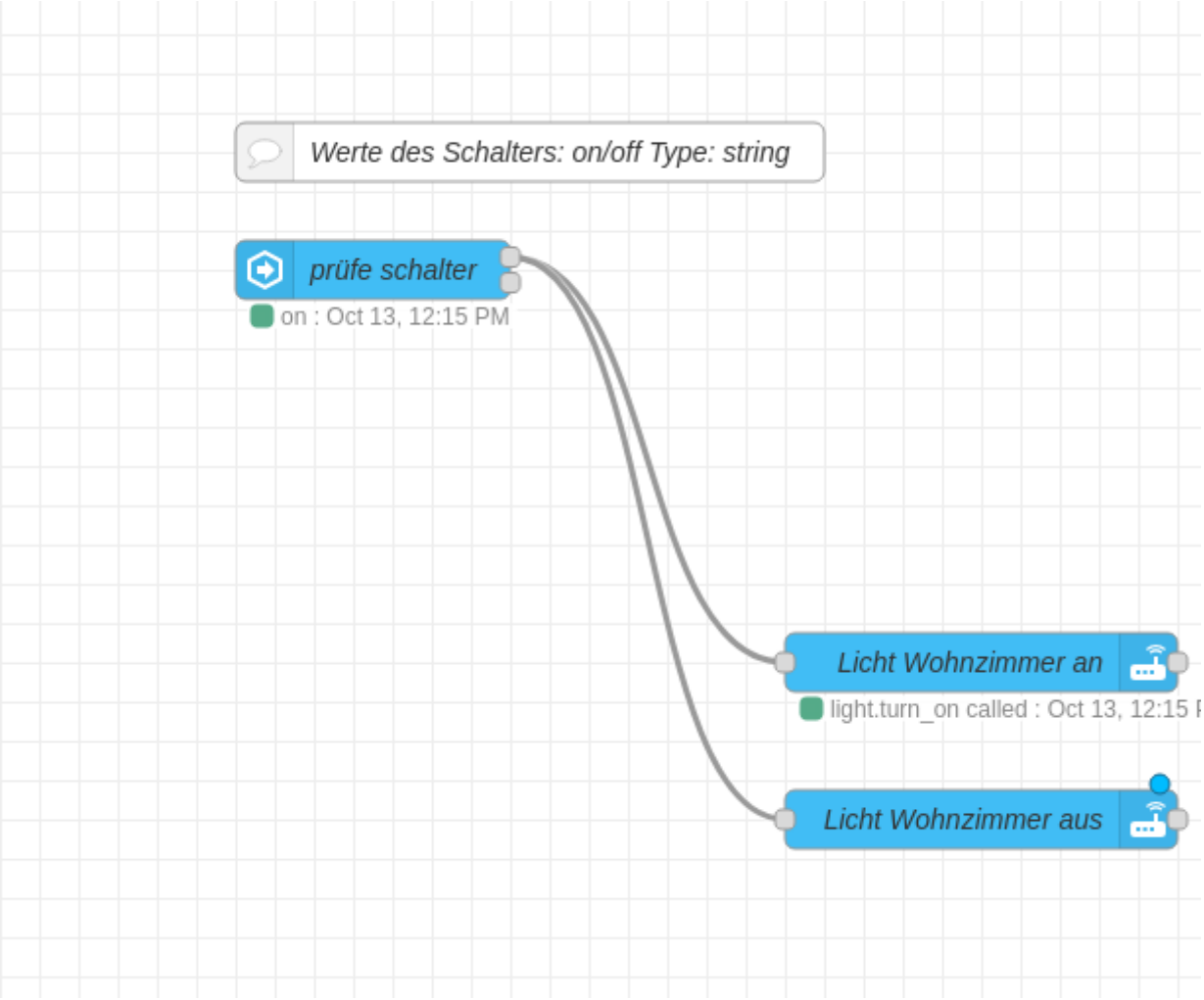
Wir klicken die erste noch verbundene doppelt an und ändern den Namen



Bei der zweiten Node ändern wir den Namen in aus und wählen den Service turn_off

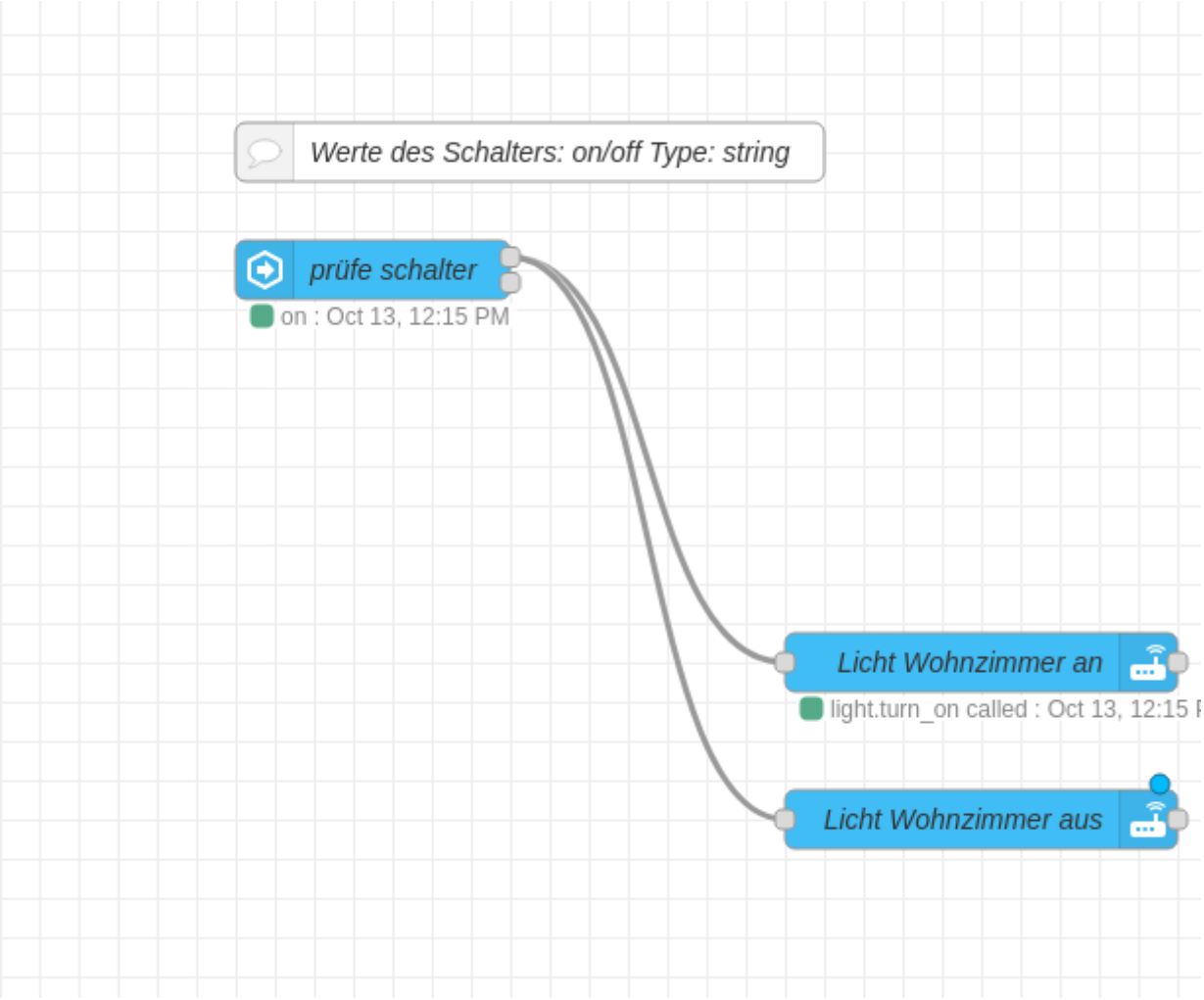


Nun verbinden wir die zweite node mit dem false Ausgang des event state, dann deployen

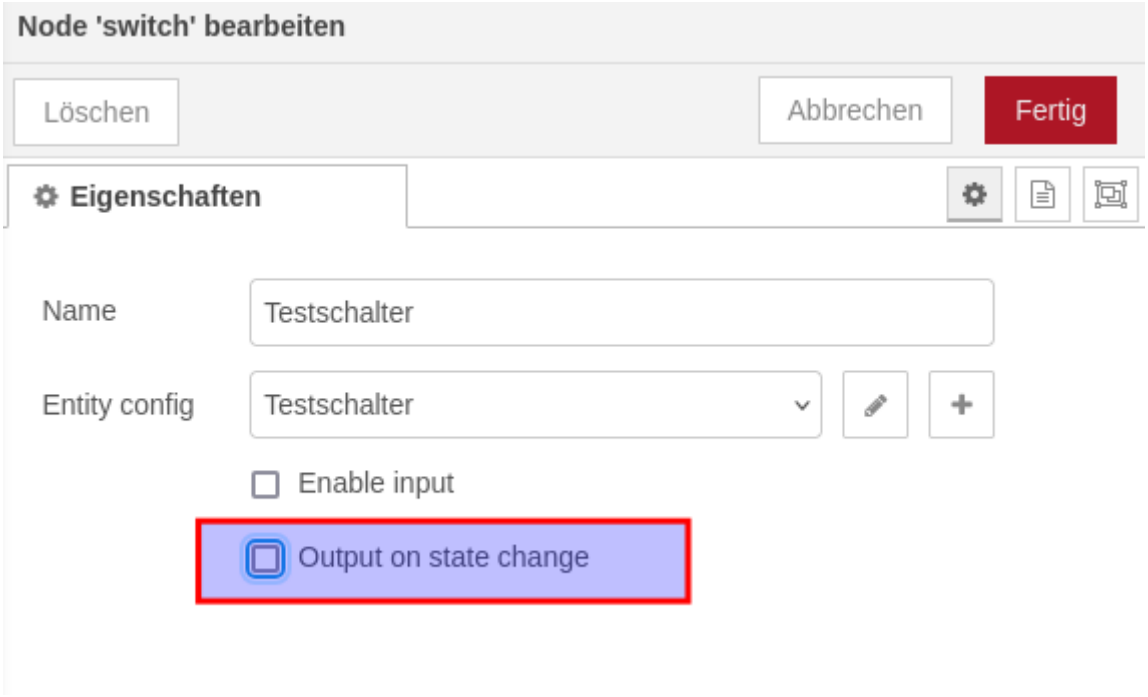


Nun kann man mittels Schalter das Licht ein und ausschalten.
Und jep Ihr habt recht, man hatte auch die Schalter Ausgänge direkt mit den Services verbinden können.
ich wollte hier aber den event state vorstellen, denn anstatt eines Schalters hätte es ja auch ein Bewegungsmelder sein können, und den können wir nur über event state abfragen.

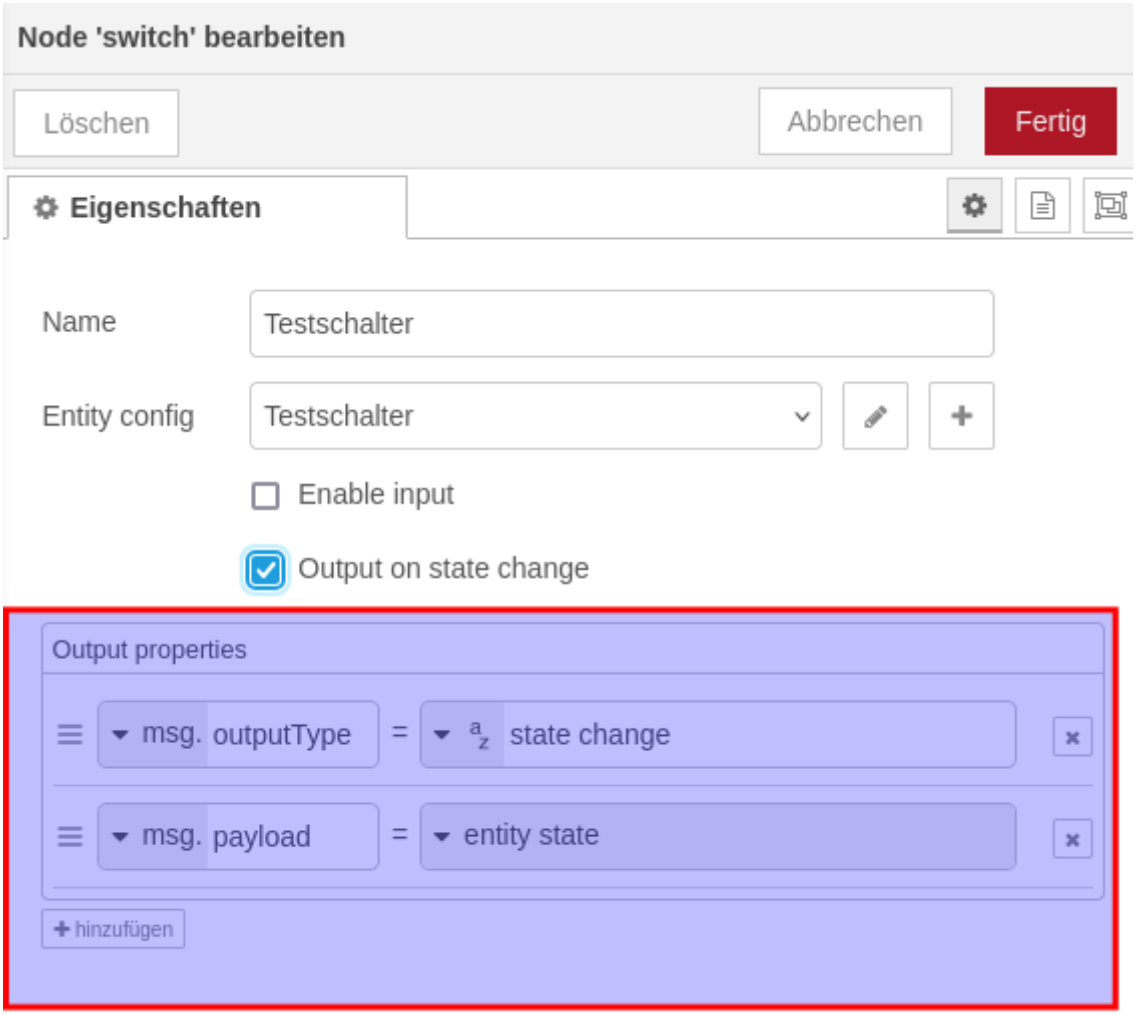
Direkt mit Schalter sähe so aus:



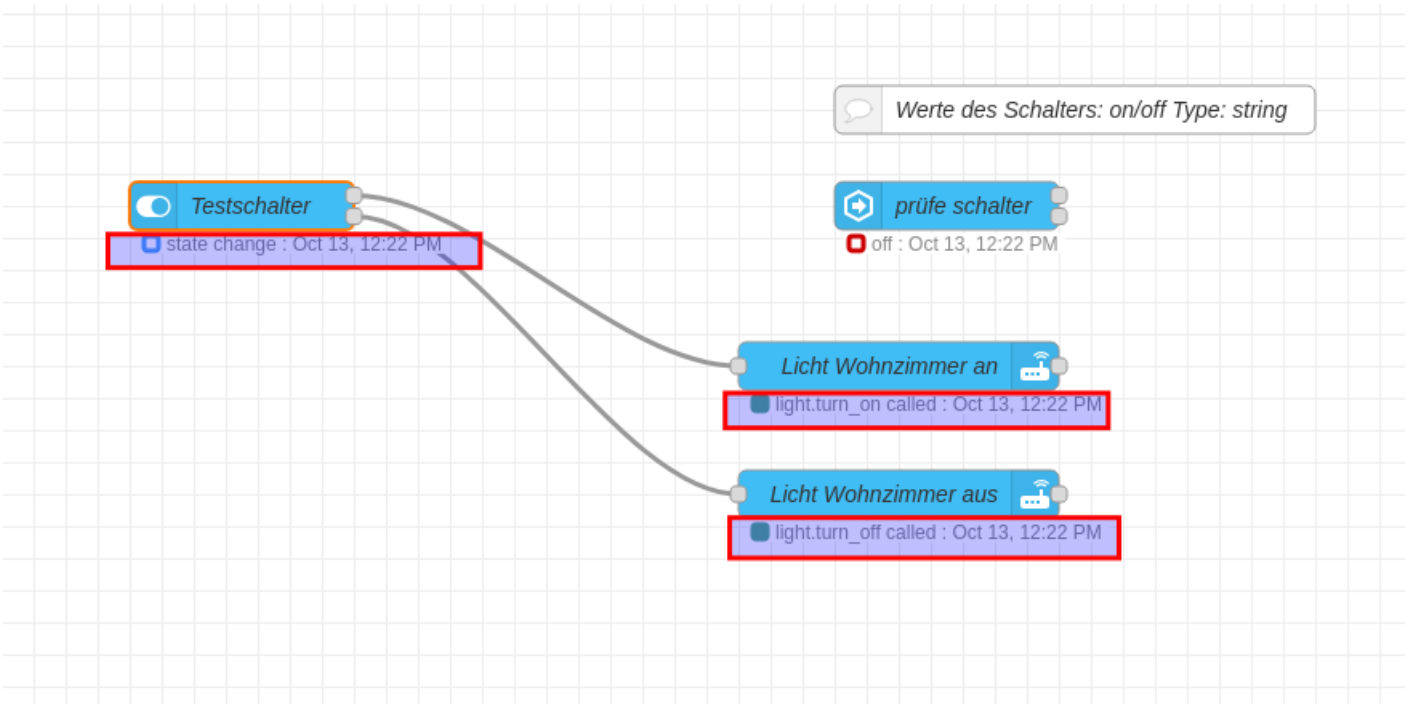
Damit das Aber klappt muss im Schalter noch ein Haken gesetzt werden das der Status als Payload weitergeben werden soll



Dies kann dann bei Standard also belassen werden



Fertig, nun kann das Licht direkt ohne event state geschaltet werden, da der Button ja schon ein event (Ereignis auslöst)
Wie hier auch nochmal zu sehen



Dieser Schalter dient wirklich nur dazu, wenn man gerne einen Knopf in der Oberfläche haben möchte der kein Echter Schalter ist.

Hat man so einen Schalter nur zum debuggen eingebaut, damit ich auch ohne event state meine Aktion testen kann, gibt es einen besseren Weg.
Denn jedes mal zum Bewegungsmelder zu rennen damit er ein Ereignis ausgibt kann ich mir einen Hilfsknopf direkt in Node Red bauen, ohne den Dashboard Knopf.
Über die Entwicklerwerkzeuge hab ich auch die Möglichkeit Werte zu ändern, aber da muss ich ja schon wieder ein Tab im Browser wechseln.

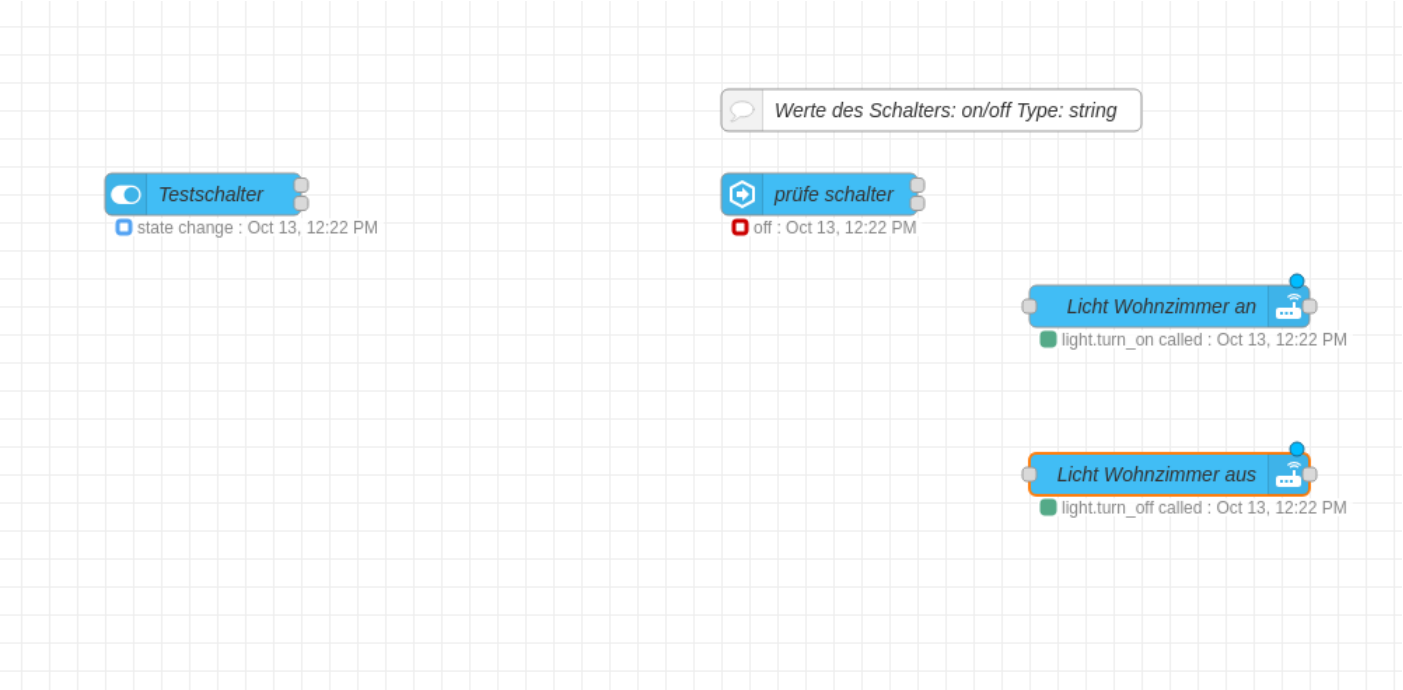
Siehe Geräte Werte Temporär ändern

Und nun der Inject Knopf

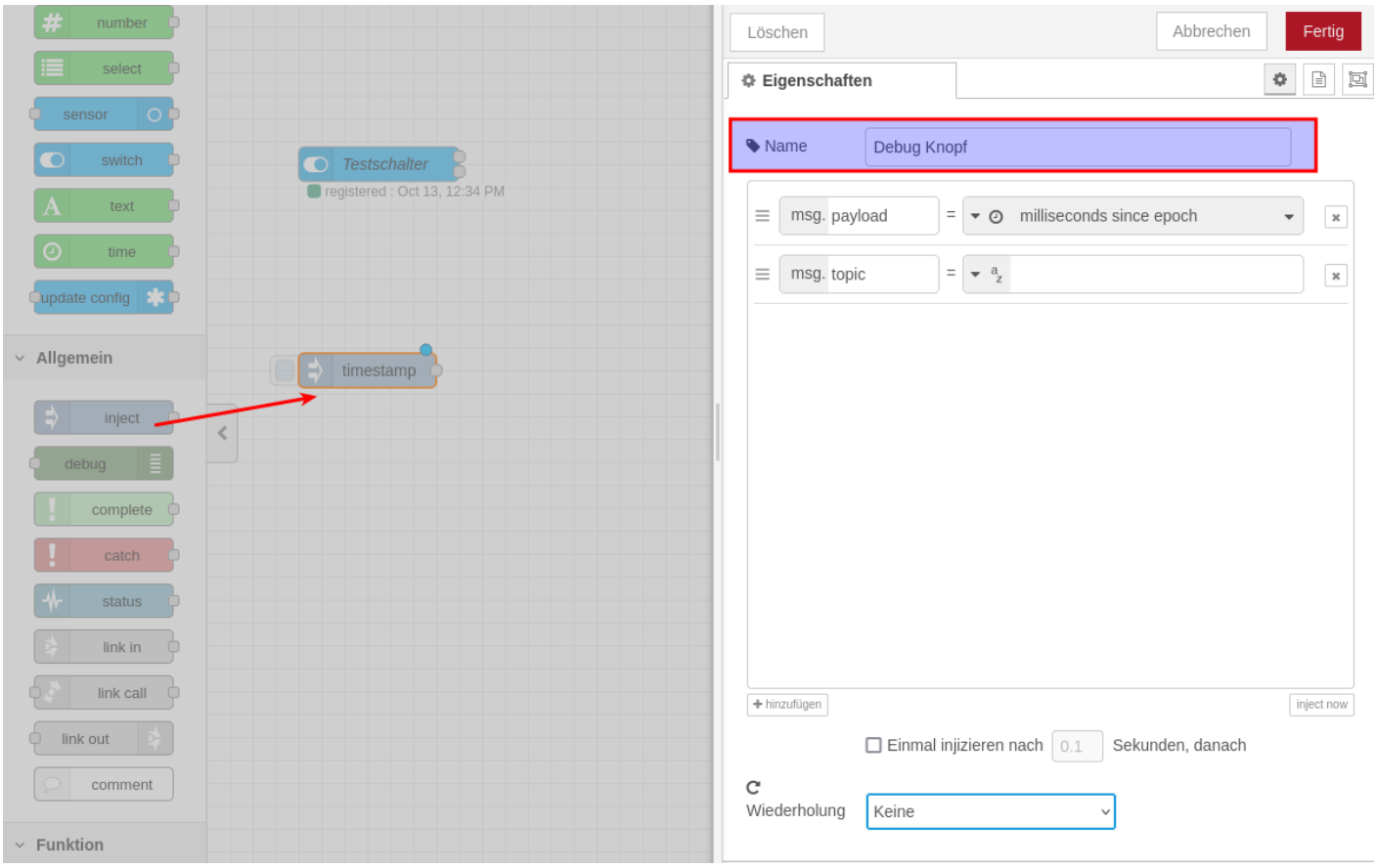
Inject Node

Der Inejct Knopf ist ein Debug Hilfsmittel mit dem Ich ein Ereignis innerhalb Node Red triggern kann.
Entweder im Intervall oder einmal.

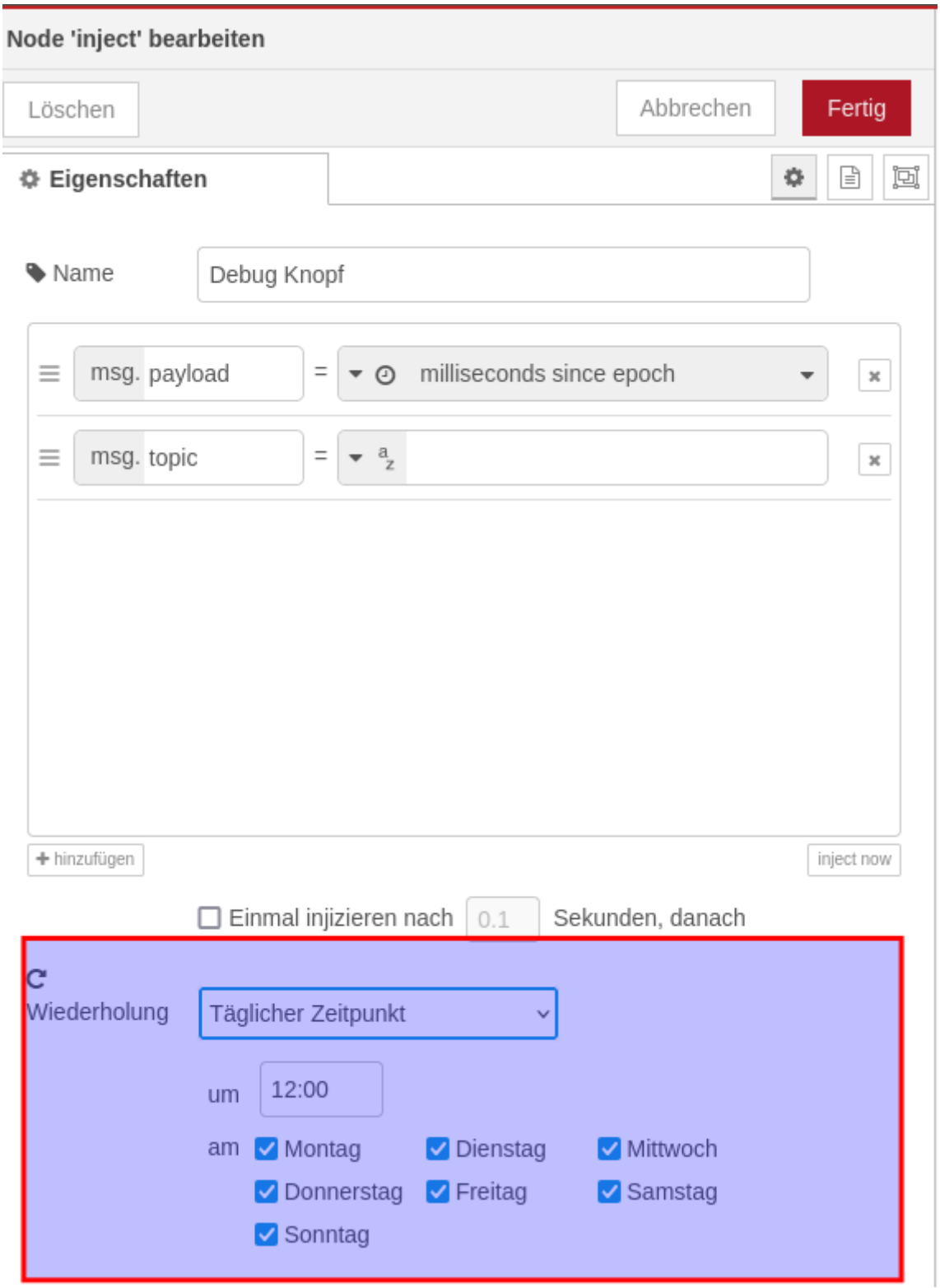
Wir löschen erstmal die Verbindungen zu unserem Vorherigen Button und schieben die Lichter ein wenig nach rechts



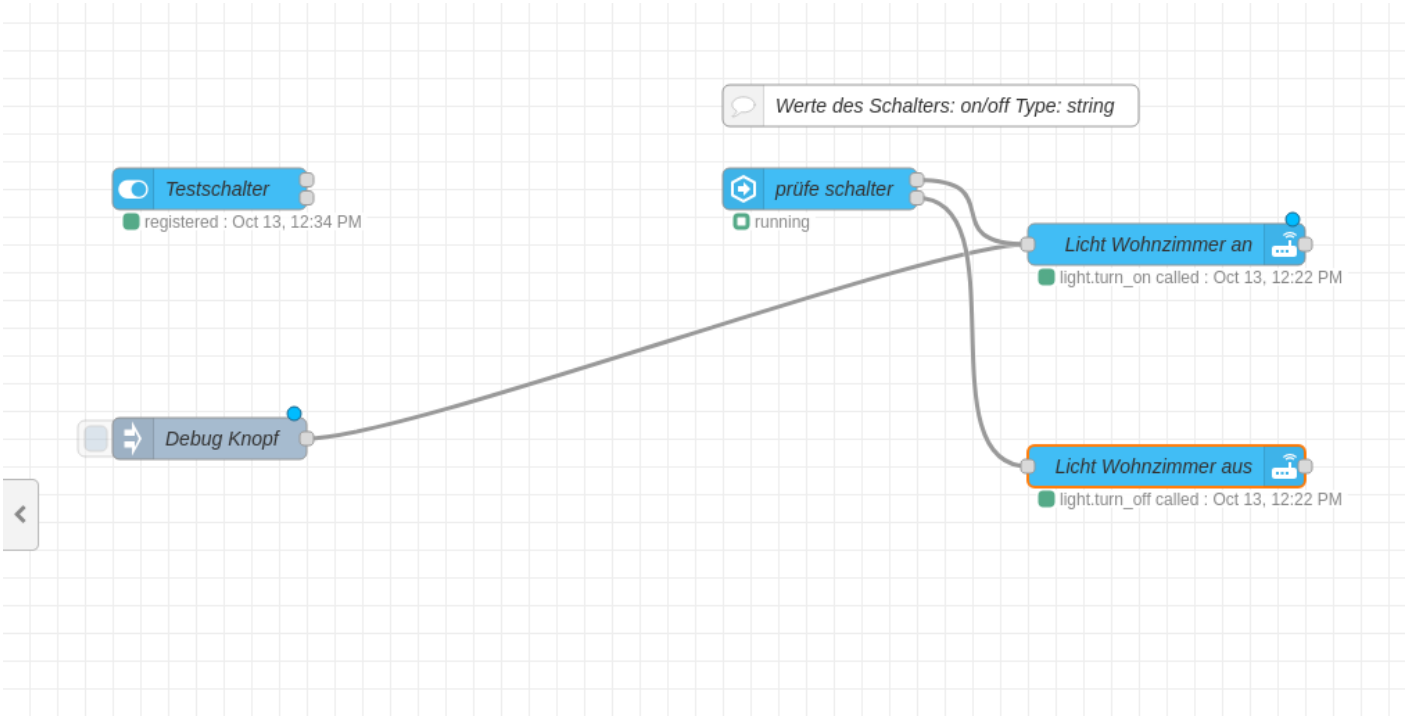
Nun fügen wir die Inject Node hinzu. Vergeben einen namen.
ALs Payload wird der Zeitstempel übergeben. Der Payload ist uns eigentlich negal, da er ja sowieso nur ein true rausgibt egal was drin steht. Das teil hat ja nur einen Zustand.



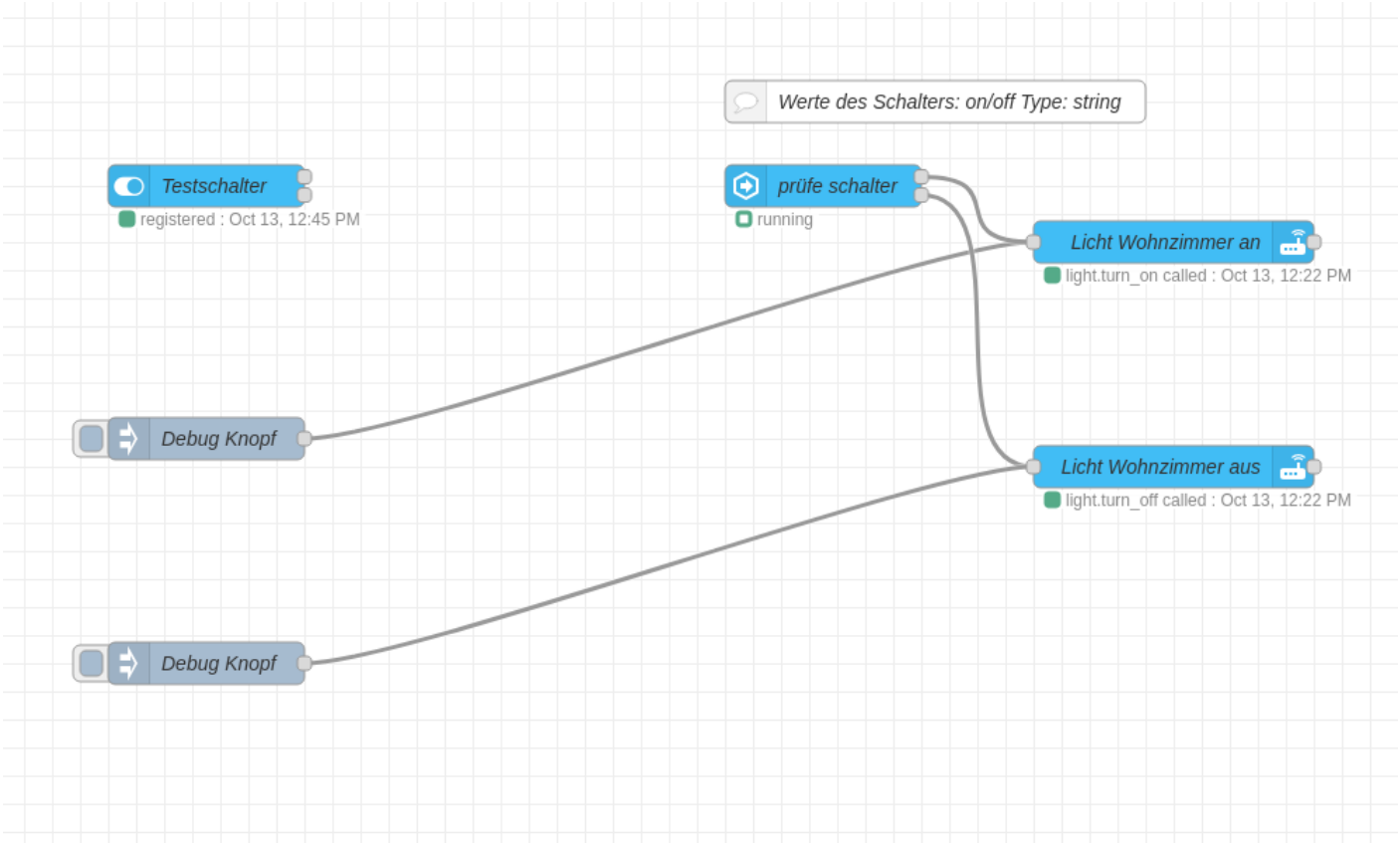
Unten wäre die Möglichkeit auch eine Uhrzeit einzusetllen, dann ist es Kein Debug Knopf mehr sondern währe ein Taskplaner.
Wie geil ist das denn ;-)



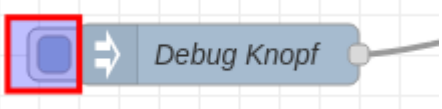
Wir nutzen Ihn hier aber nur als Debug Button zum Auslösen beim drauf drücken.
Diesen verbinden wir jetzt mit Lmape an. Genauso wie unseren Prüfe schalter Verbinden wir auch wieder mit Lampe an und Lampe aus.
Dann sieht das ganze wieder so aus. Wird der Dashboard Schalter gedrückt löst das event state wieder aus jenach nzustand an oder aus. Wird unser Debug Knopf gedrückt wird das licht angeschaltet unabhängig vom event state



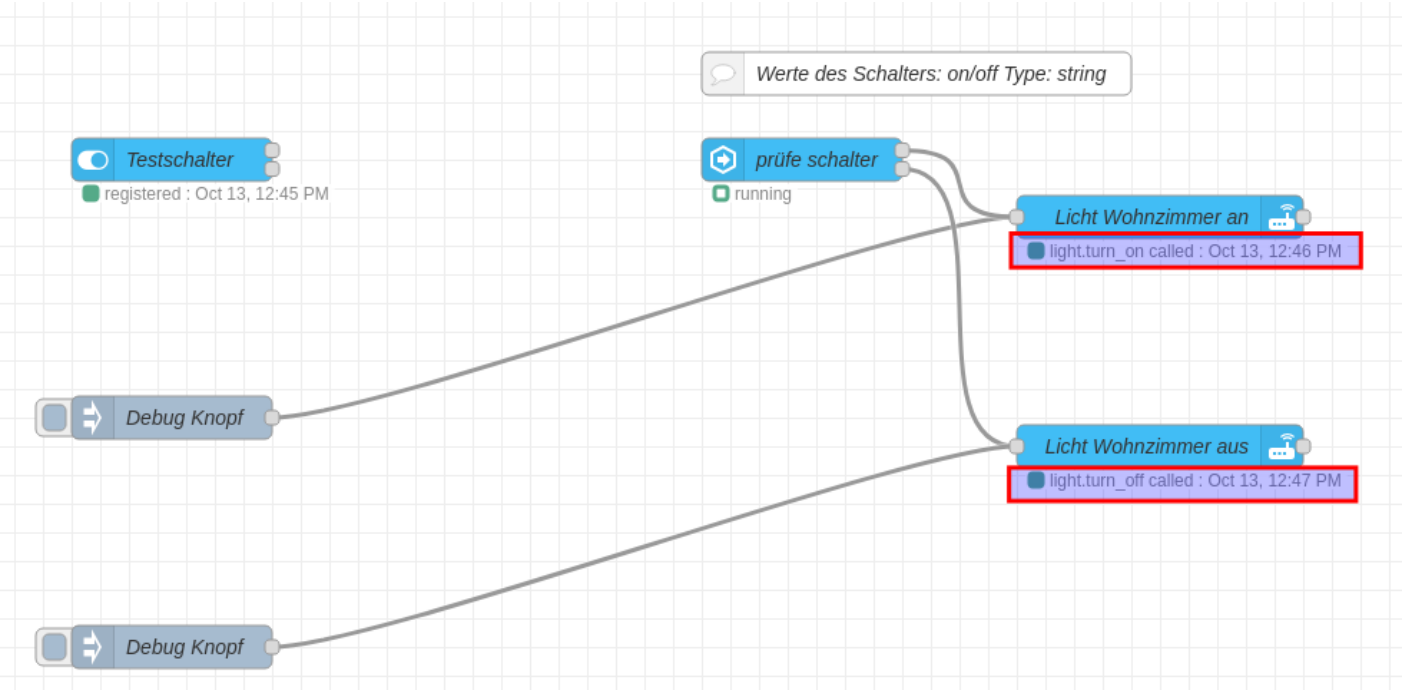
Wir kopieren den Debug Knopf noch einmal und verbinden diesen mit Wohnzimmer aus und deployen das ganze.



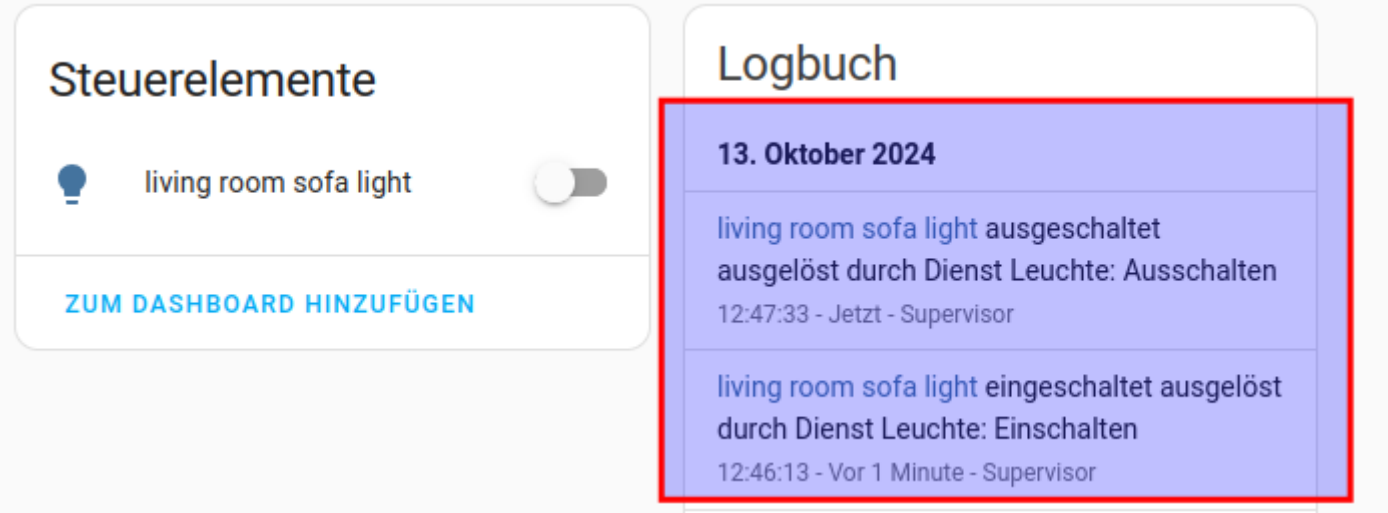
man löst den Debug knopf über die Schaltfläche davor aus, wie bei der Debug Node den Knopf



Anhand der Uhrzeit sieht mann ich hab erst on dann off gedrückt.



Hier auch nochmal im Log der Lampe



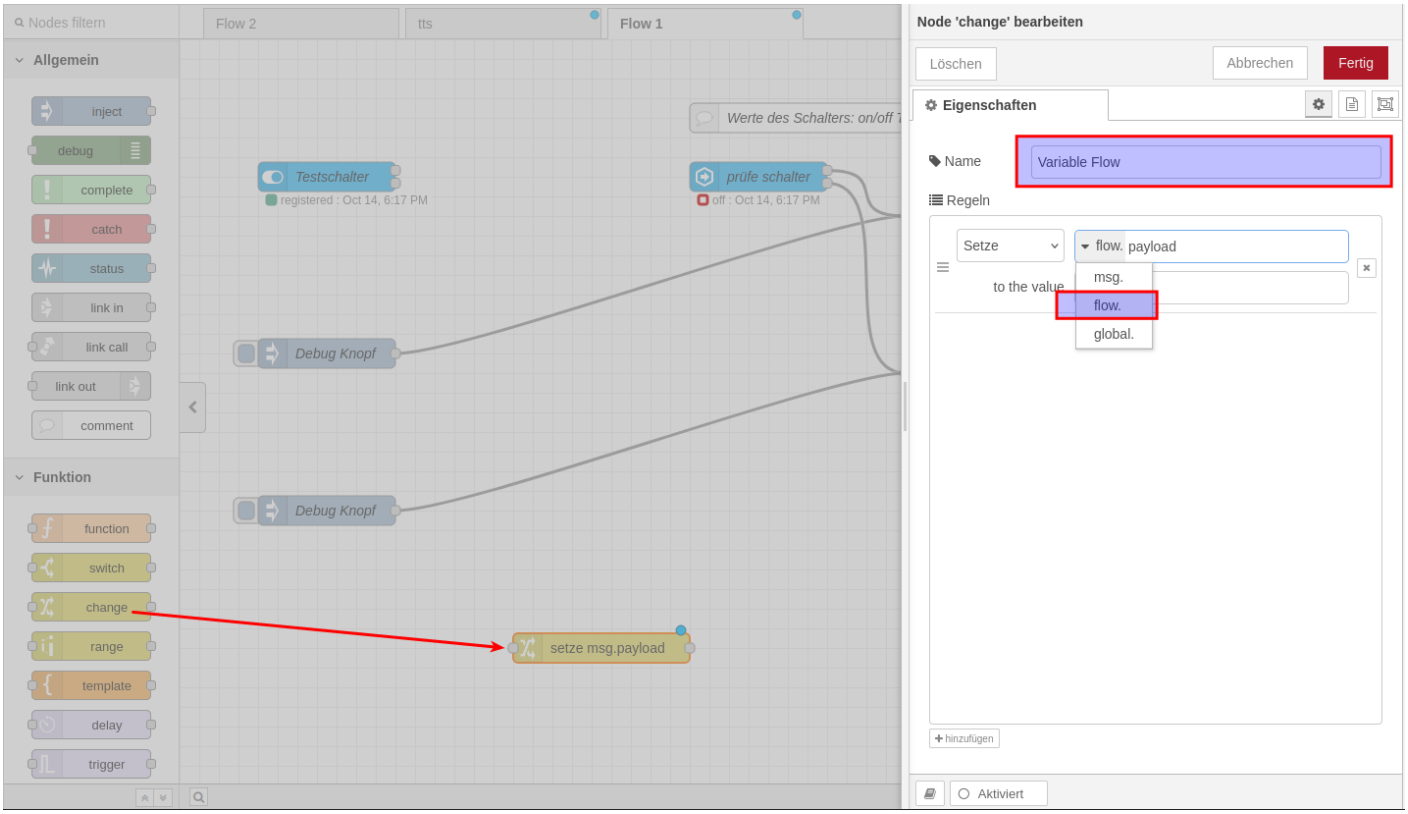
Anstatt den Button in der event statse noder zu überwachen, würde man hier einen Bewegungsmelder nehmen oder was auch immer nehmen, ist schon klar.
Es geht hier ums Prinzip ;-)

Variablen: Change und Function Node :

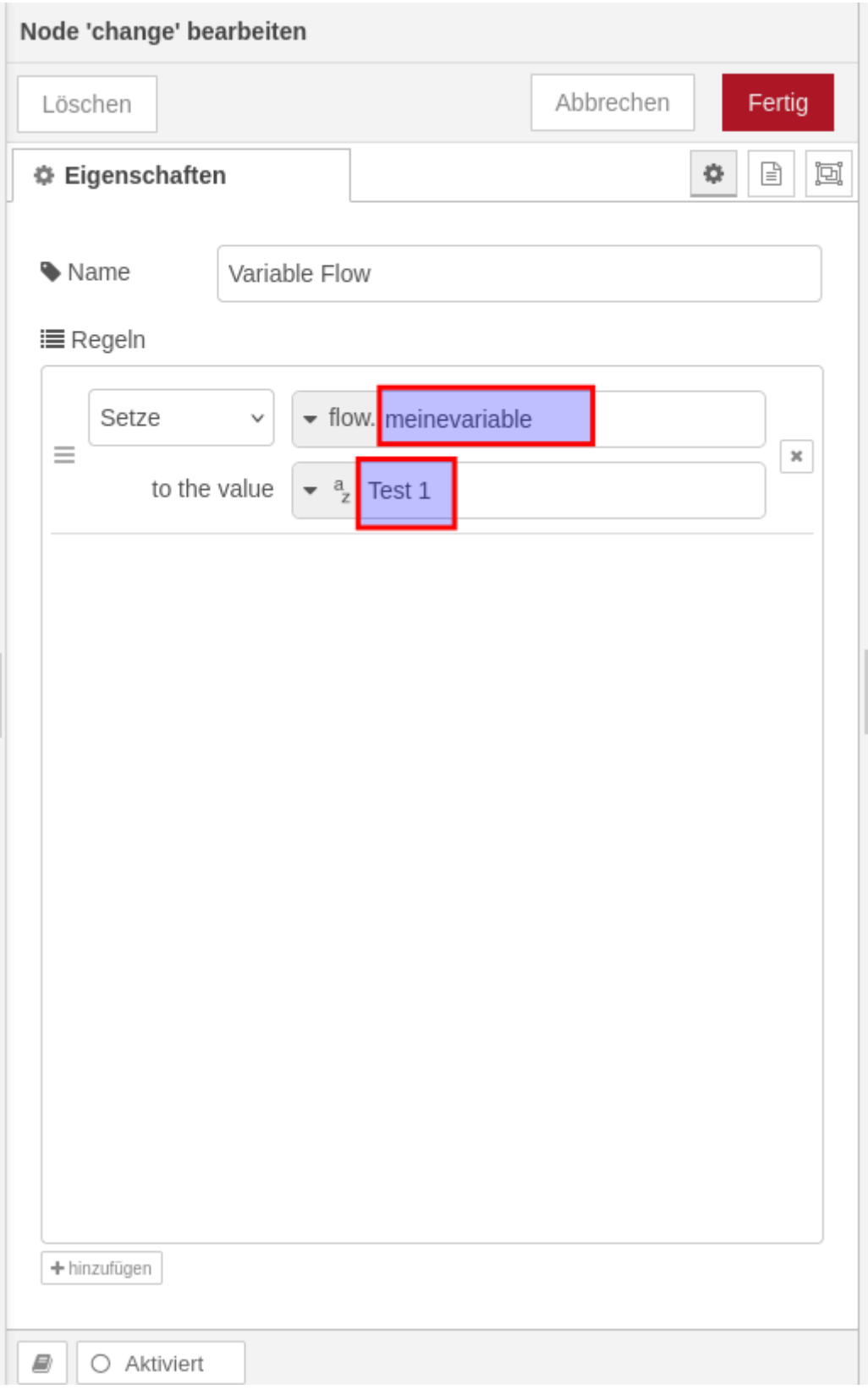
Es gibt zwei Typen von Variablen:

1. Flow Variablen - Gelten nur in dem aktuellen Flow
2. Global Variablen - Gelten über alle Flows

Dazu ziehen wir uns eine Change Node in den Flow, setzten den Namen.
und wählen dann flow oder Global aus der Liste aus.

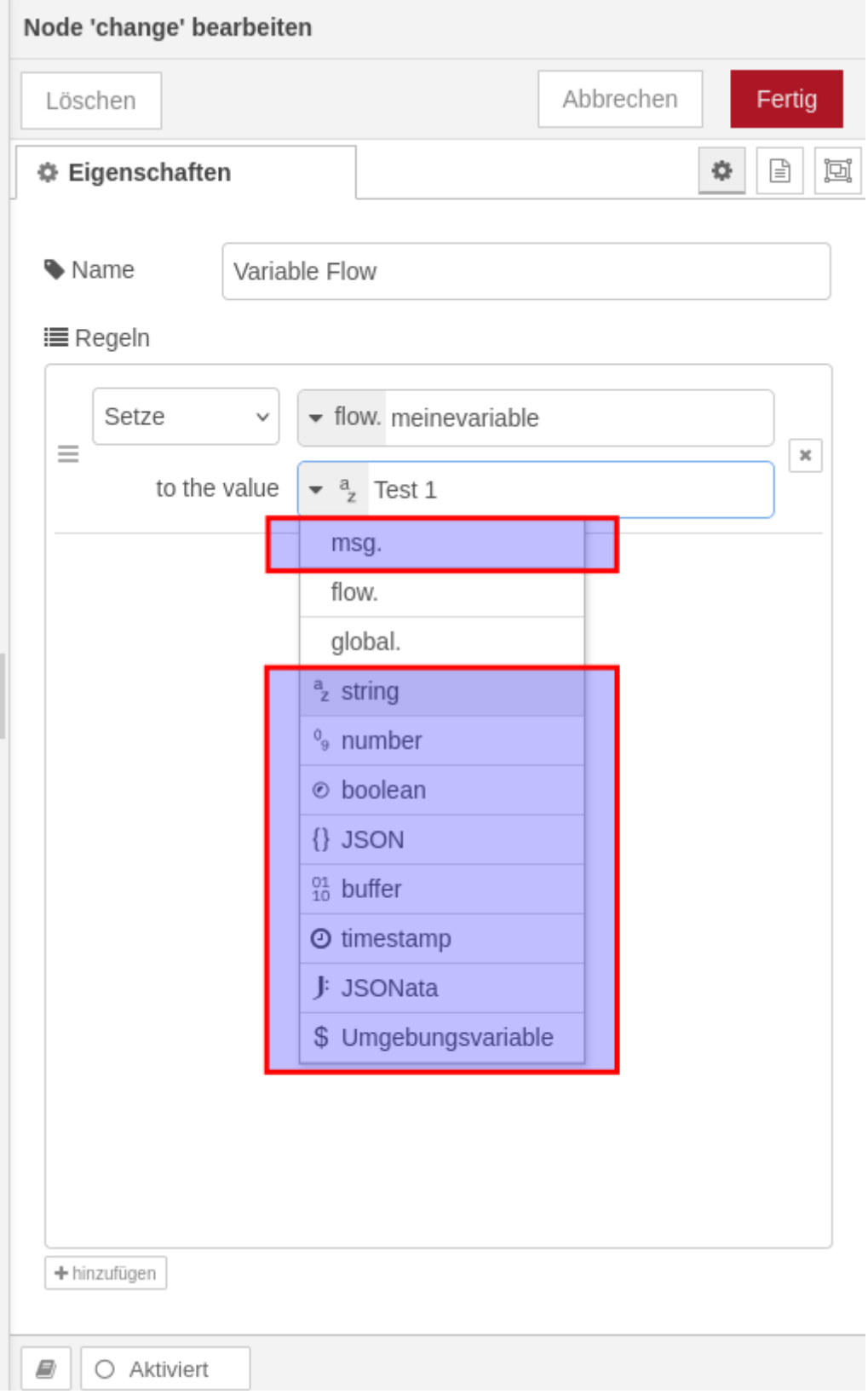


Nach flow. den variable Namen und darunter der Wert

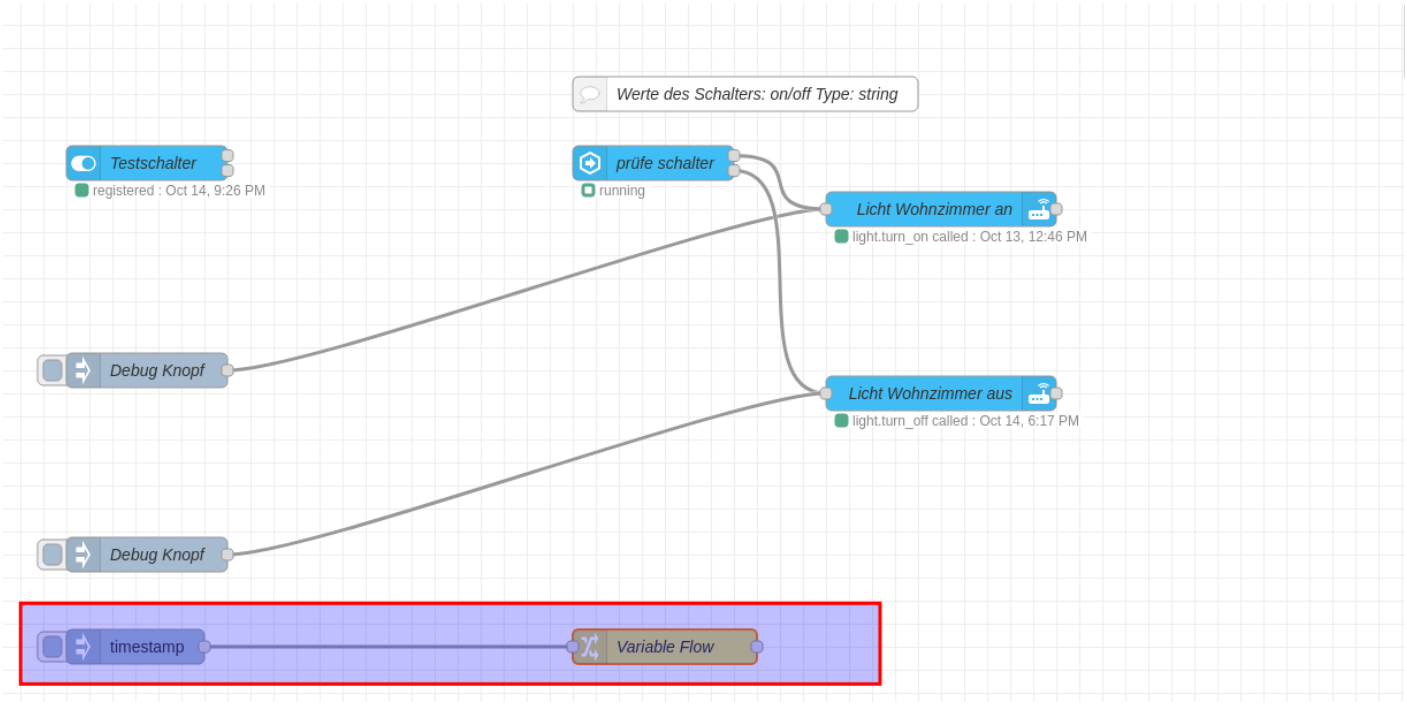


Uns stehen folgende Variable Typen zur Verfügung. Flow und Global machen meist keinen Sinnd weil wir hier ja eine flow/global Variable definieren. Möchten wir eine bestehende Variable ändern, kommt meist noch mehr dazu als nur Inhalt Variable 1 in Variable 2 kopieren
(was natürlich geht in dem to Value dann die Flowvariable ausgewählt wird mit dem Quellvariablennamen).
Für komplexere Sachen nehmen wir lieber dann die später erklärte Function Node.

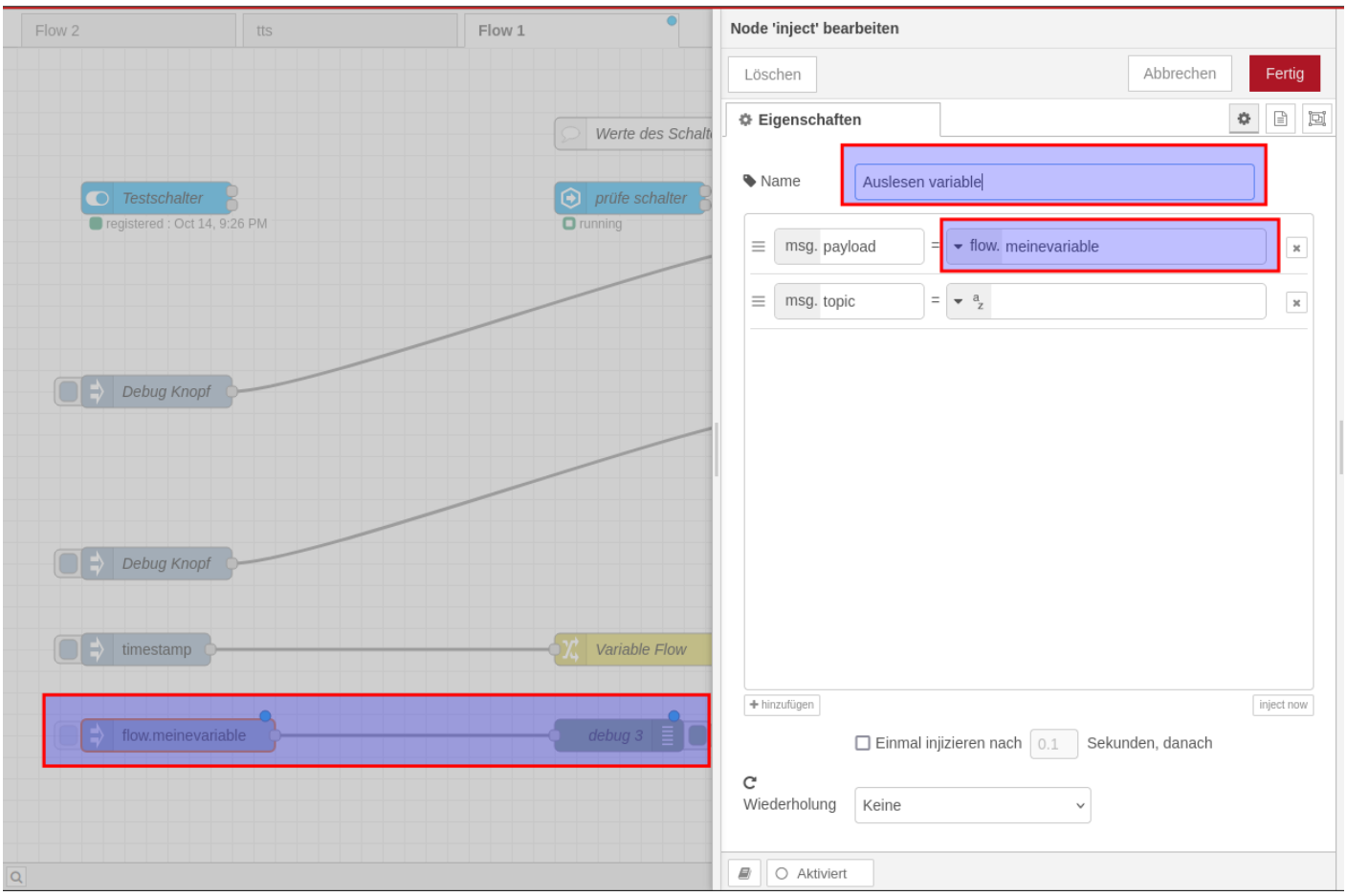
Das bei Set to value ist das msg Objekt, das msg was am Eingang der Node ankommt zum Beispiel der Wert eines vorherigen verbundenen Sensors oder so.



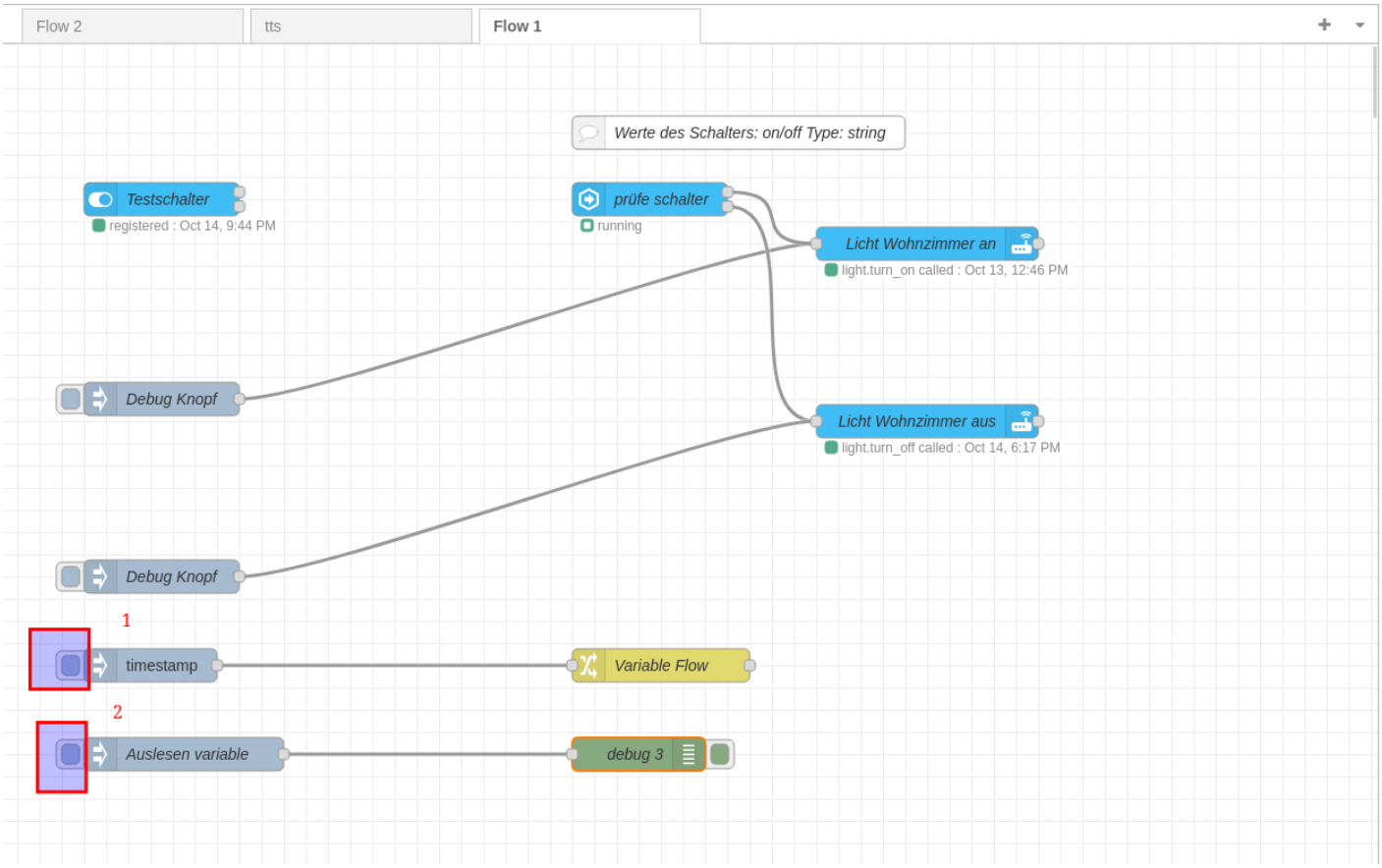
Nun setzten wir uns ein Inject Knopf und verbinden die Change Node mit um diesen dann triggern.



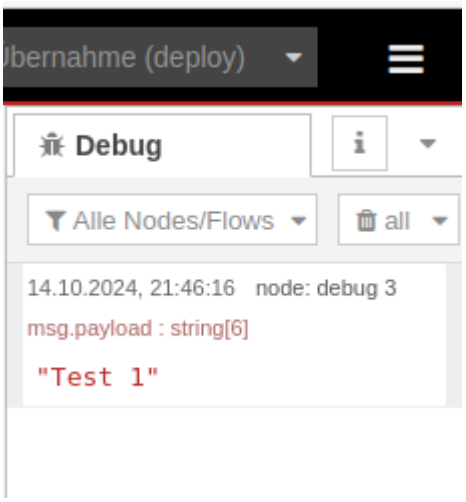
Setzen uns eine weiteres Inject und Verbinden eine Debug Node.
Dort einen Namen wählen und als payload flow und unseren Variable namen.



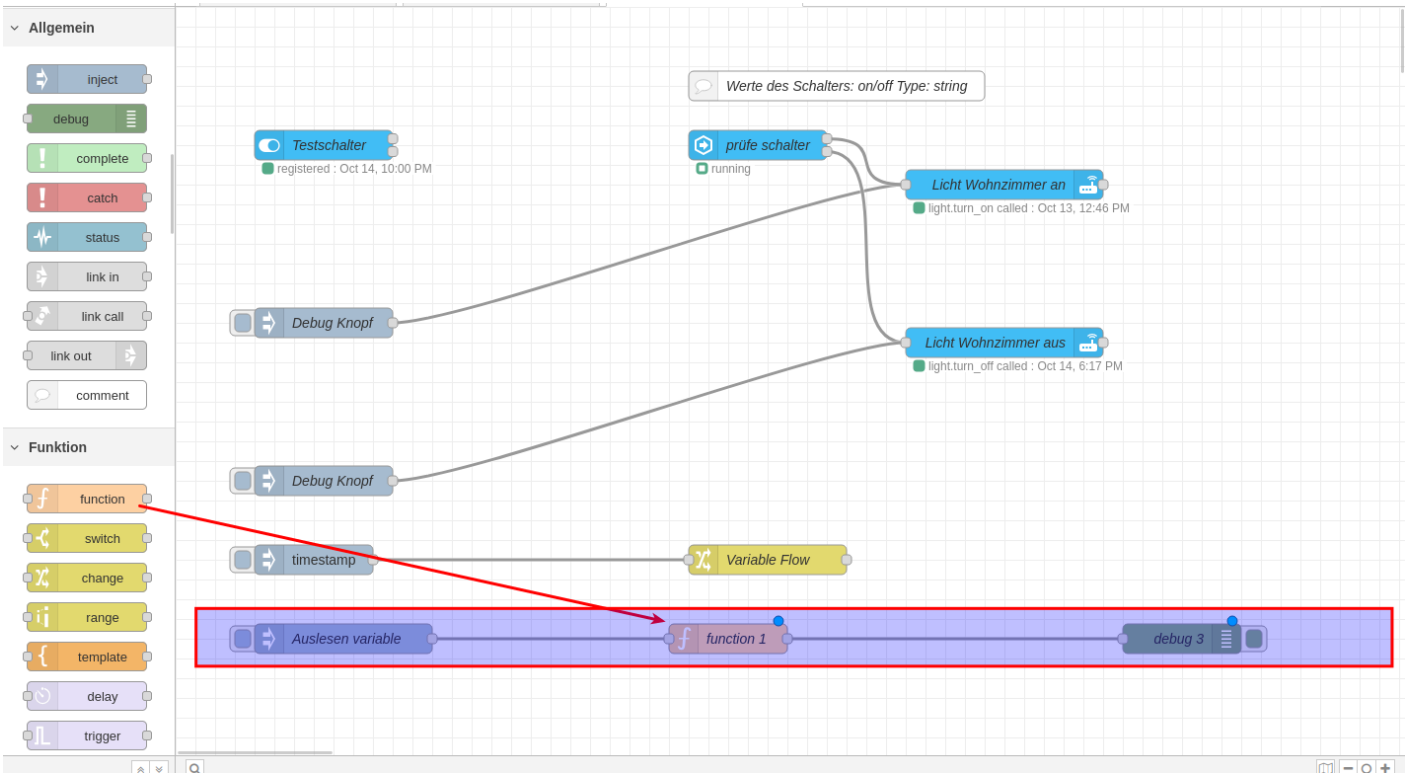
Nun deployen wir das ganze und drücken dann auf den inject button 1 und danach inject button 2



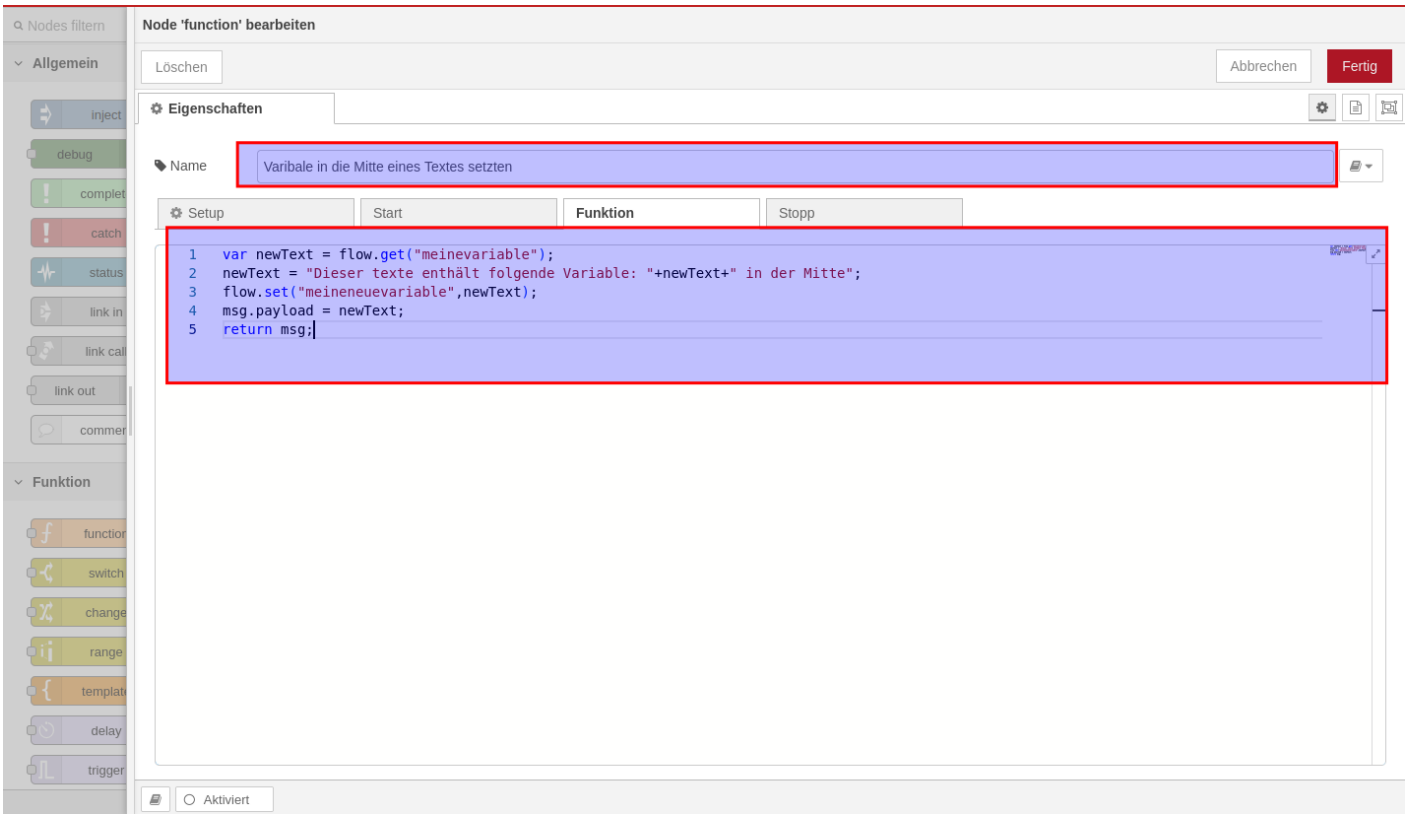
Nun sehen wir in der Debug leiste unseren Wert.



Wir können die Variable aber auch innerhalb eines Payloads nutzen also sprich als Varibale in einem text oder so.
Dazu nutzen wir eine Function Node und setzten diese zwischen dem inject und dem debug



Doppelklick auf die Function und die Function anpassen



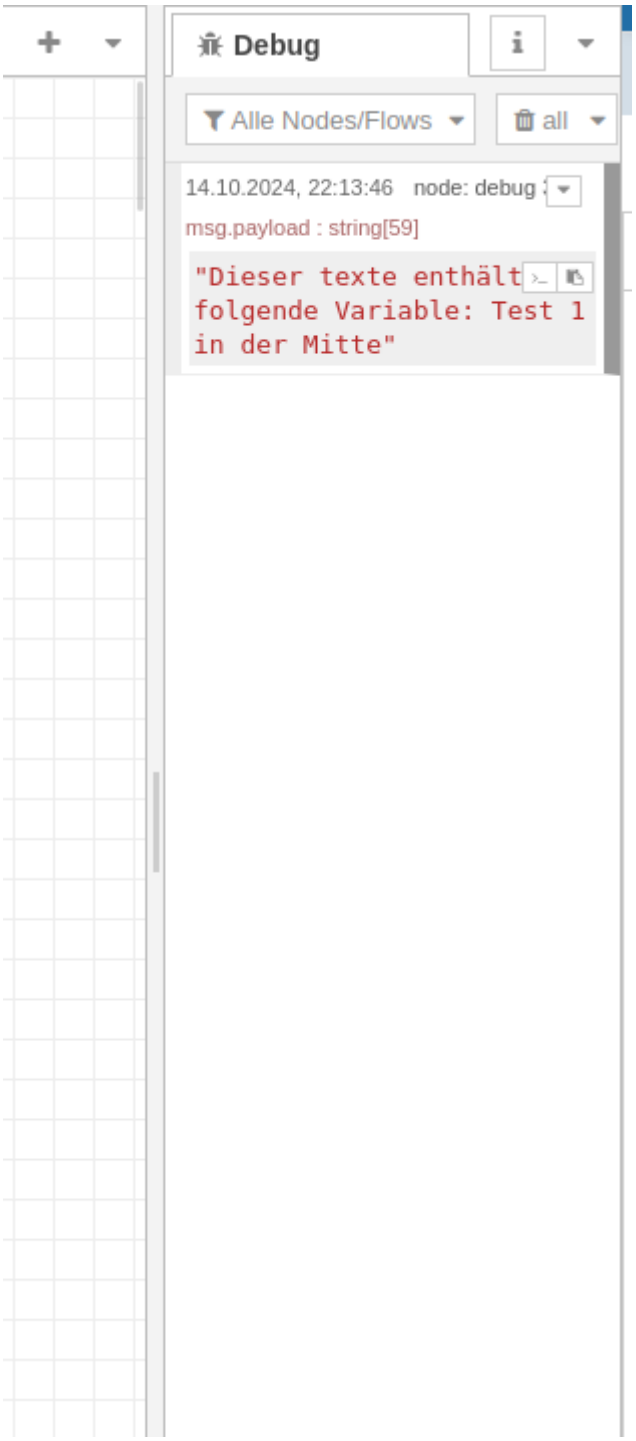
Beschreibung:

var newText = flow.get("meinevariable"); //wir holen die flow Variable meinevariable und stecken sie in eine loakle in dieser funktion

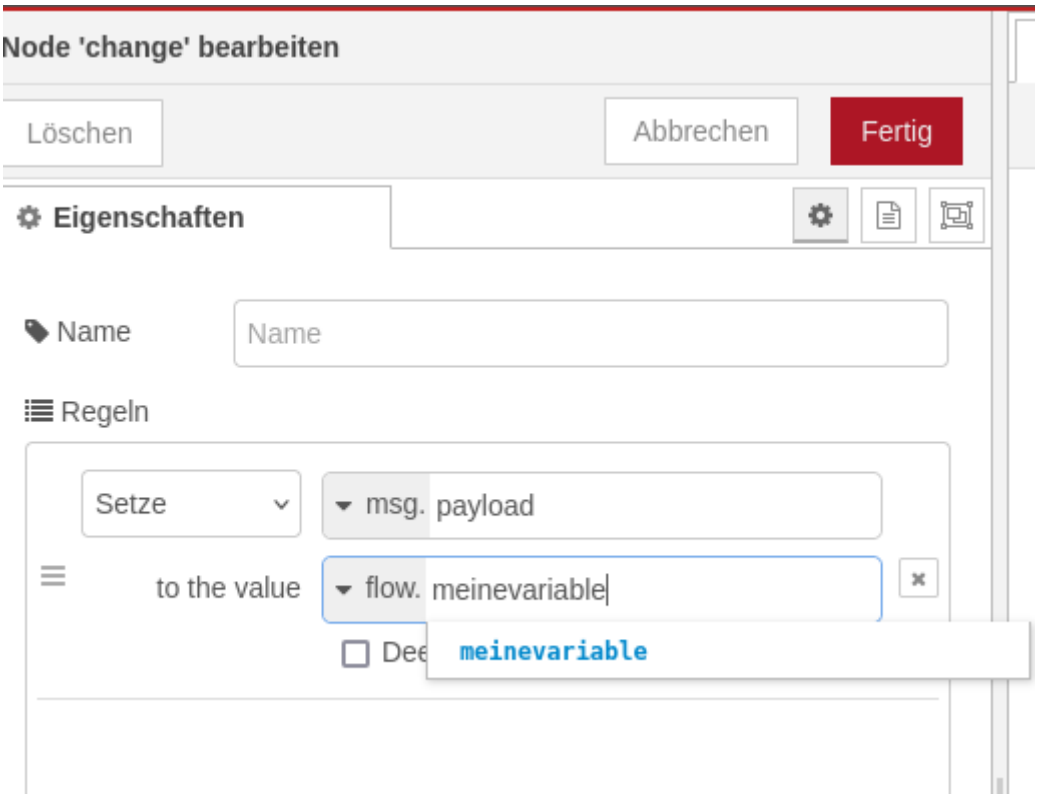
newText = "Dieser texte enthält folgende Variable: "+newText+" in der Mitte"; //wir überschreiben newtext mit

```
einem text und in der mitte die ausgelesene Variable die noch in newtext steckt.
//newtext ist jetzt der neue text
flow.set("meineneuevariable",newText); //wir legen eine neue flow variable an, falls man darauf später zurgeifen möchte und nicht gleich in der nächsten node
msg.payload = newText; //gleichzeitig geben wir als payload den neuen text aus, falls man gleich schon mit dem payload in der nächsten node weiterarbeiten möchte, man kann aber auch die neue flow variable nehmen
return msg; //die message zurückgeben
```

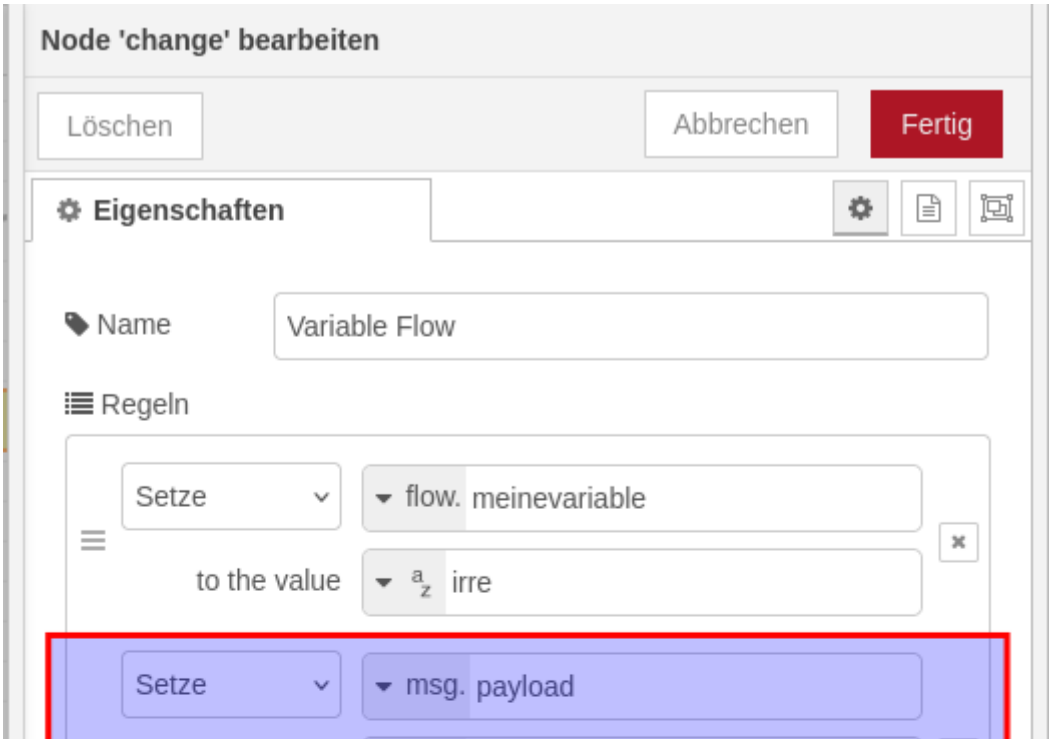
Da wir als Payload auch unseren neuen Text ausgegeben haben, ist der auch sofort im Debug Node sichtabr



DIe flow.meineneue Variable können wir auch in einer weiteren function, weiterverarbeiten oder mit einer change node die msg.payload einfach auf den Wert des flows/global setzten
Beispiel Eigenschaften einer change node.
Der Name muss natürlich auch ausgefüllt werden, geht ums Prinzip, die mgs payload auf den Inhalt eines flows/global zu setzten:



Eine Function macht dann sinn, wenn man einer Variable Werte hinzufügen/ rechnen oder was was ich mach möchte oder gleich mehrere dinge auf einmal.
Wie die Variable komplex ändern und gleichzeitig als Payload ausgeben.
Will man aber tastächlich nur variable definieren und als payload ausgeben geht das auch mit change. denn mit hinzufügen kann man mehre aktionen durchführen.
Diese werden von oben nach unten abgearbeitet.
Beispiel:



Wir können aber auch anstatt eine Change Node zu nehmen auch gleich eine Function Node nehmen um dieses durch zu führen, eine Variable zu definieren und als Payload ausgeben. Allerdings finde ich zum reinen definieren ne Change Node einfacher.
Das gleiche geht natürlich dann auch mit global anstatt flow global schreiben, wenn man global variablen braucht.

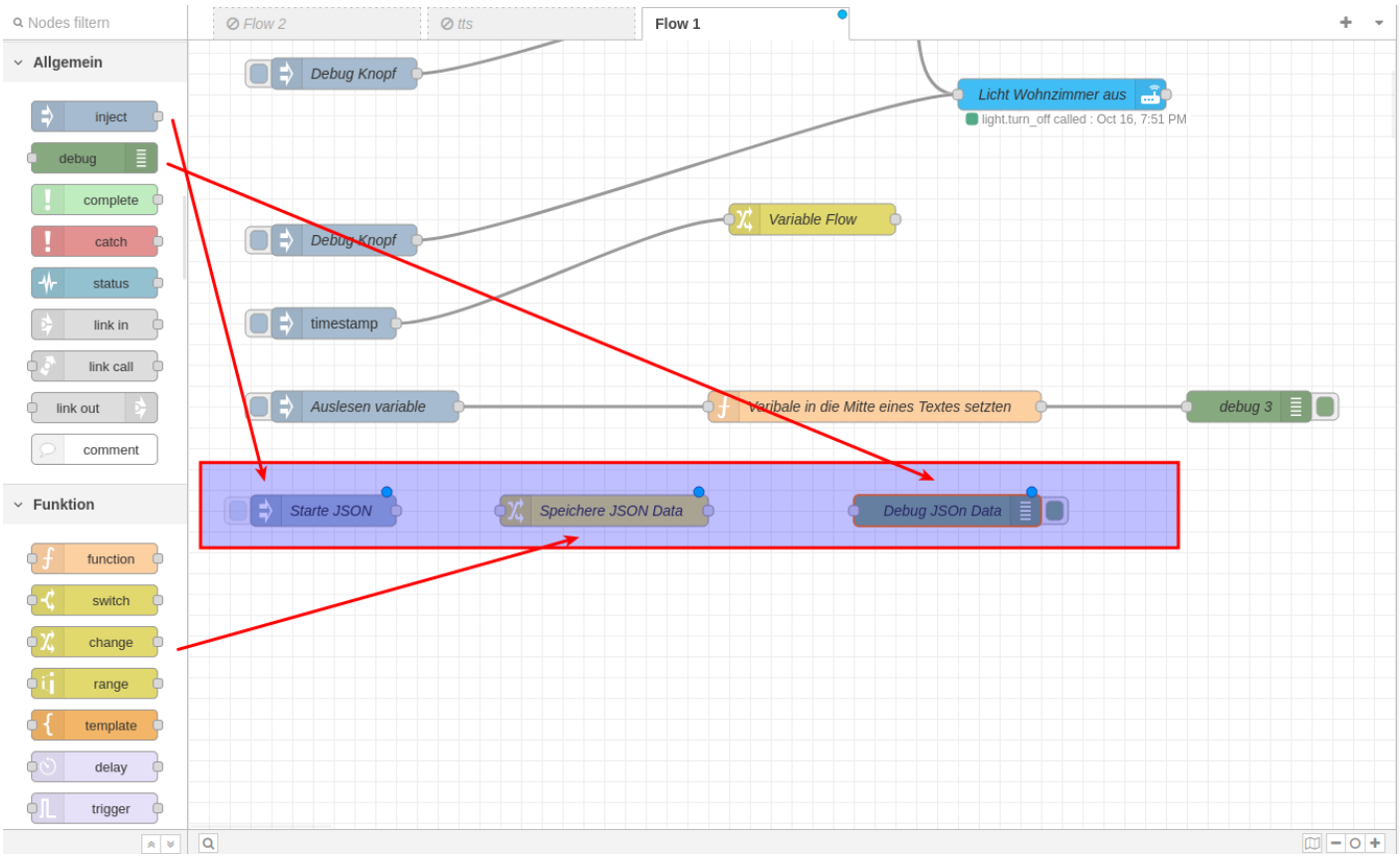
JSON in eine Variable laden, so hat man ein Array in einer Variablen:

Wie man Variablen anlegt haben wir ja schon gesehen.
So speichern wir ein JSON Object in einer Variablen und lesen einzelne Elemete wieder aus.
So können wir mehrere Werte in einer Variablen speichern. Und hätten damit ein Array

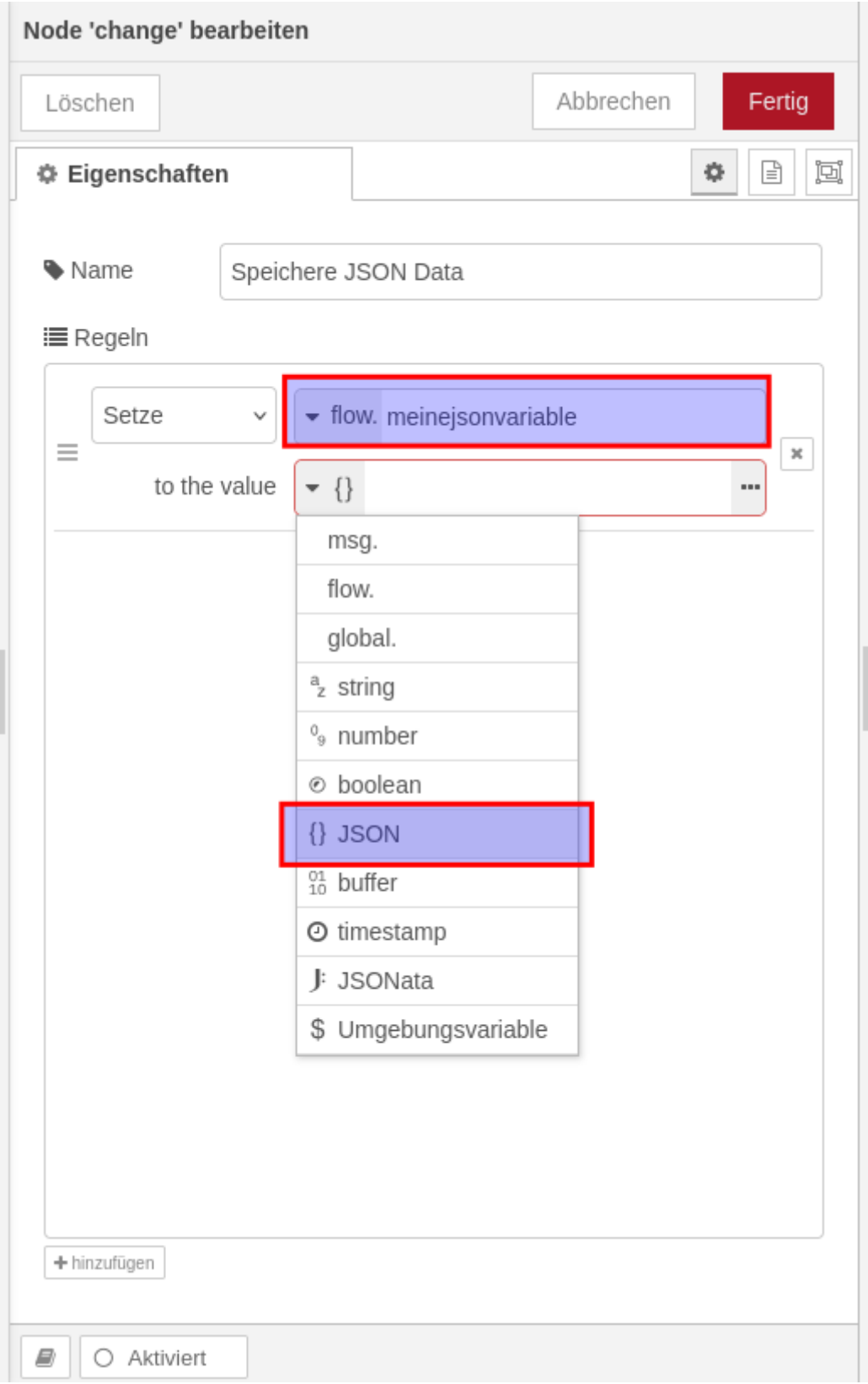
Hier mal ein Beispiel JSON. Hier haben wir 4 Eigenschaften: name, color,on,brightness

```
{
"name":"licht Sofa",
   "color":"blau",
   "on":"true",
   "brightness":100
}
```

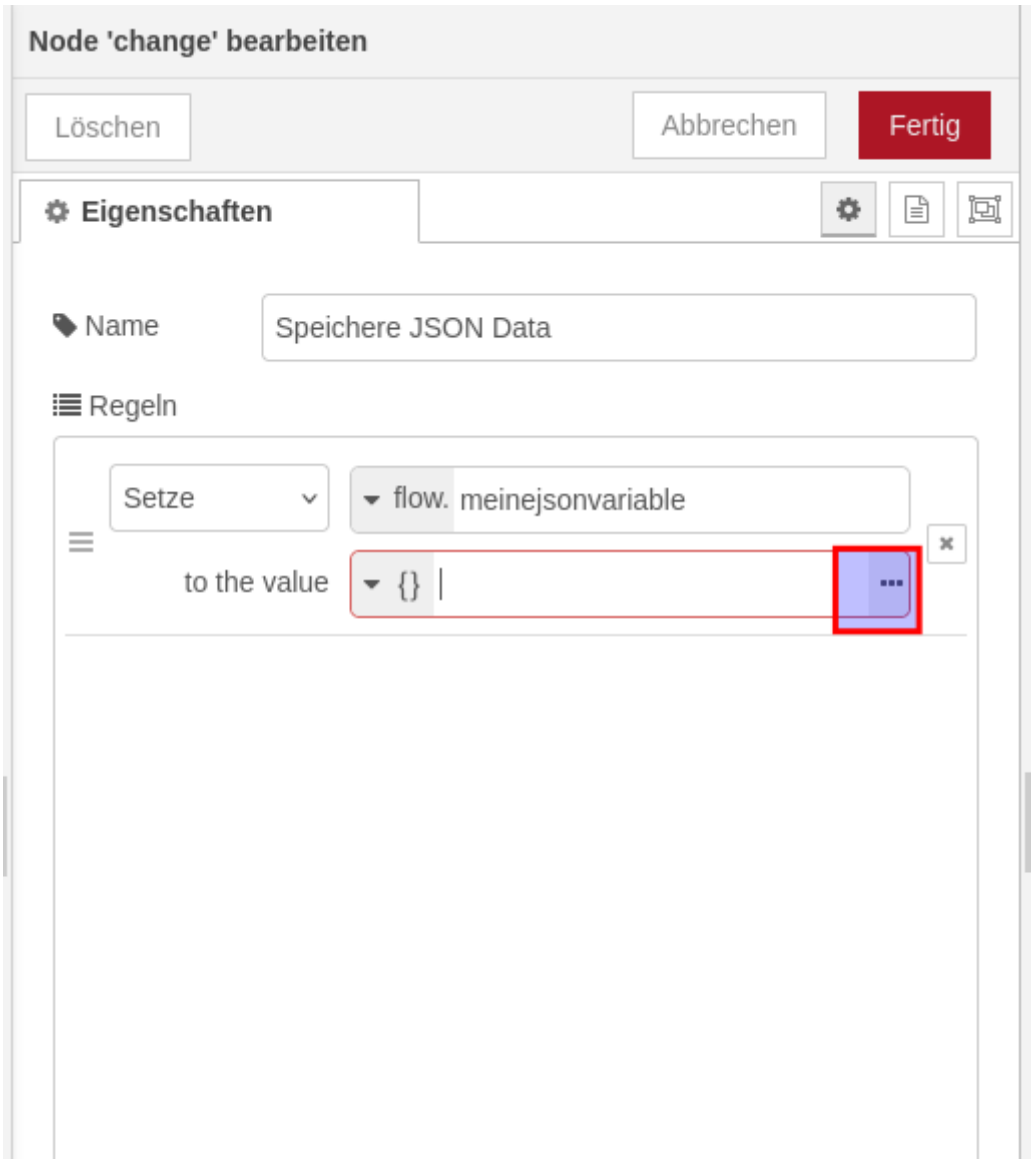
Mit einer Change node können wir diese JSON speichern. Kommt sie von einer msg nutzen wir ebenfalls die change node.
Wir erstellen einen inject button eine change node und eine debug node



Doppelklick auf die Change Node
Dort vergeben wir einen Namen, bei setze wieder flow/gloabl und den variablennamen und bei set to wählen wir json aus.

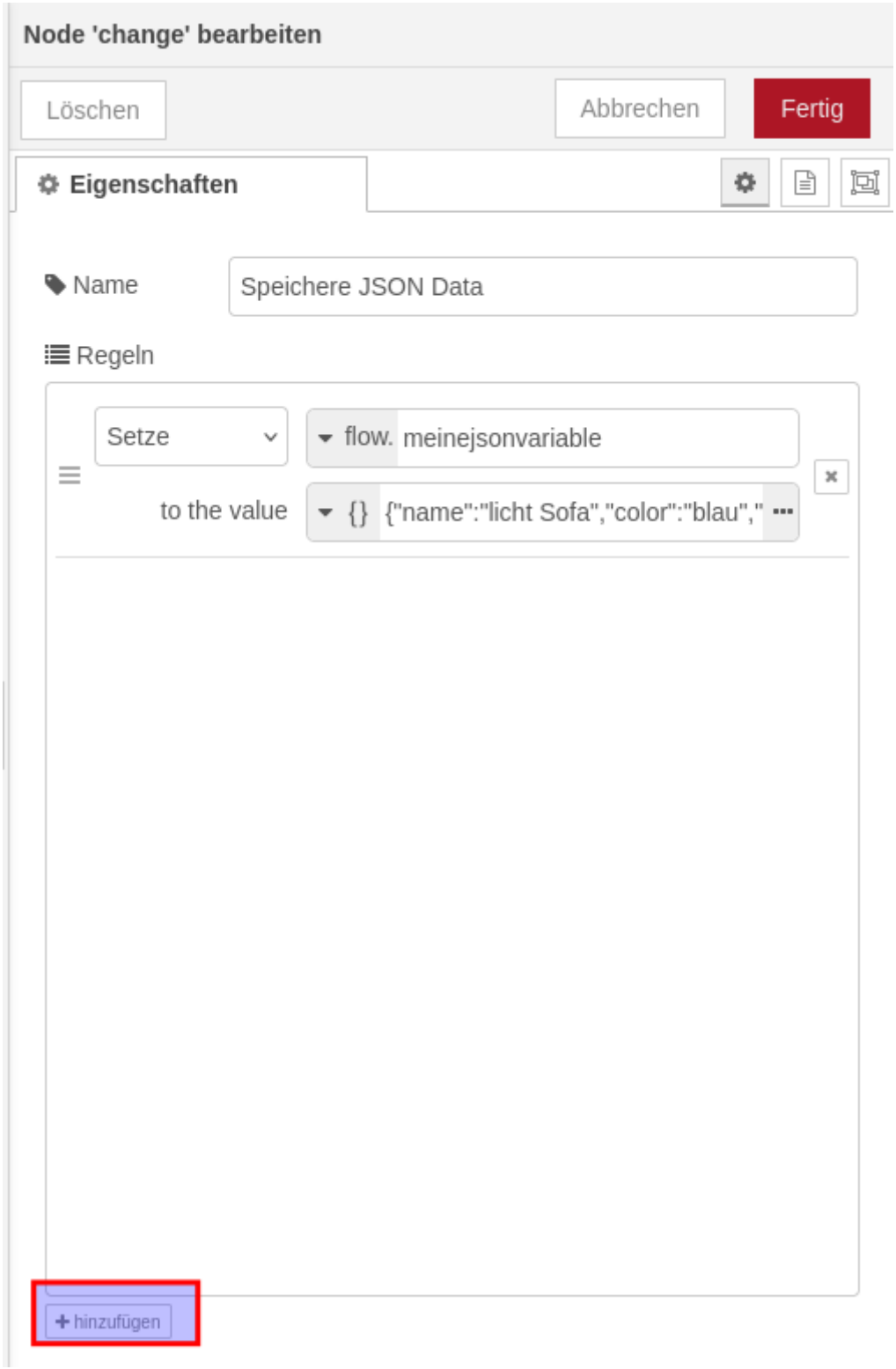


nun klicken wir auf die 3 Punkte

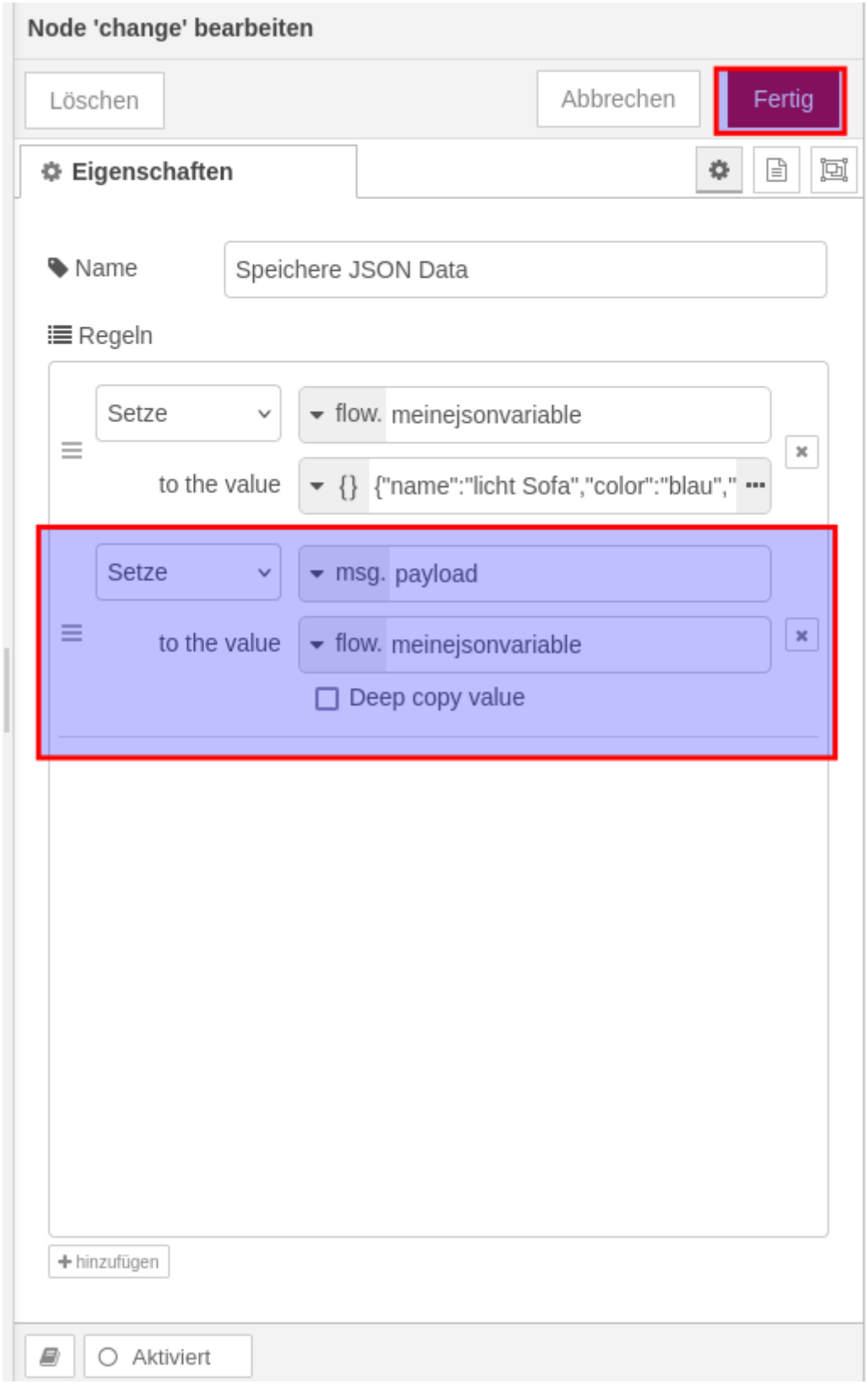

Node 'change' bearbeiten
Löschen
Abbrechen
Fertig
Eigenschaften
Name
Speichere JSON Data
Regeln
Setze
flow.
meinejsonvariable
to the value
{}

und fügen unser json Data Beispiel von oben ein und dann auf fertig

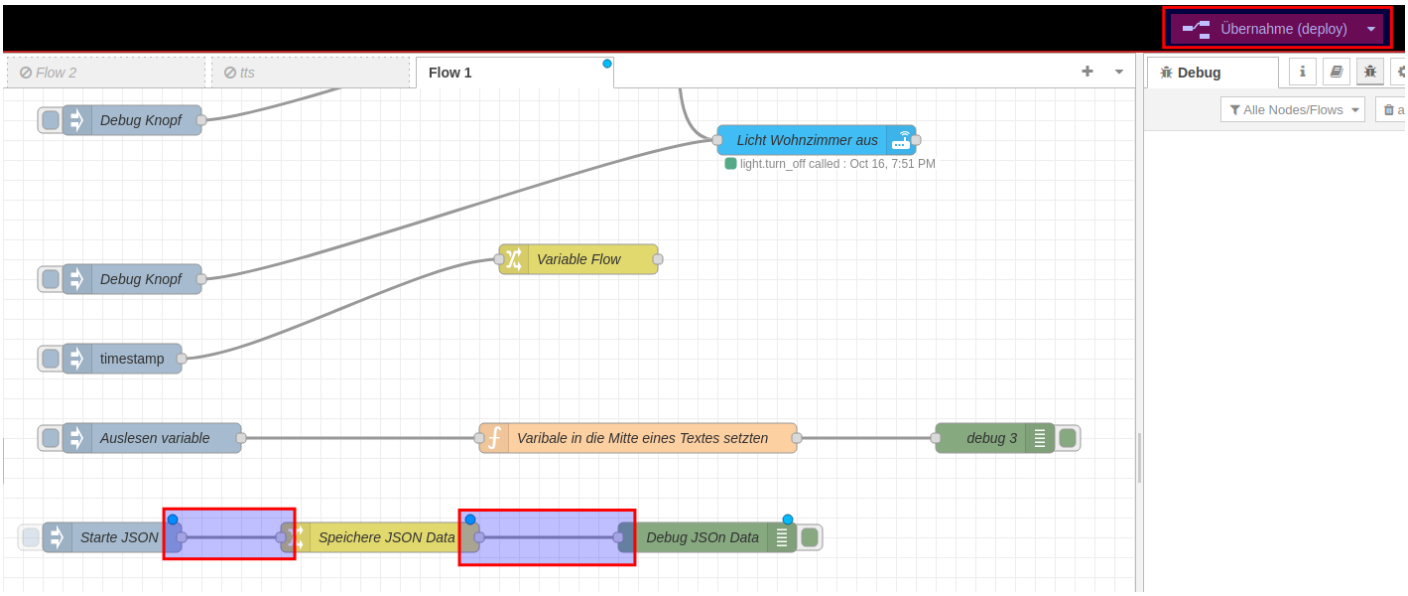
Nun sieht das ganze so aus und jetzt unten auf hinzufügen klicken



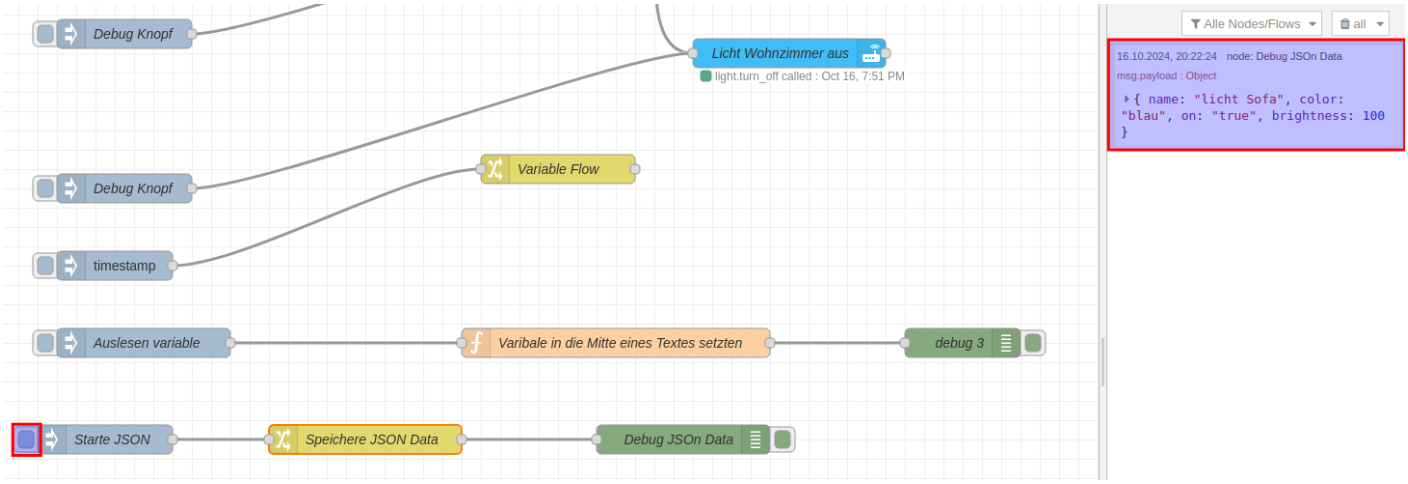
Nun kann die msg.payload Ausgabe gesetzt werden auf die flow Variable die wir im ersten schritt defniert haben, danach auf fertig. Diese variable kann nun in der nächsten node durch setzten des msg.payload, wie auch später über den Variablenamen wieder verwendet werden.



Nun unsere Nodes Verbinden und dann deploy anklicken

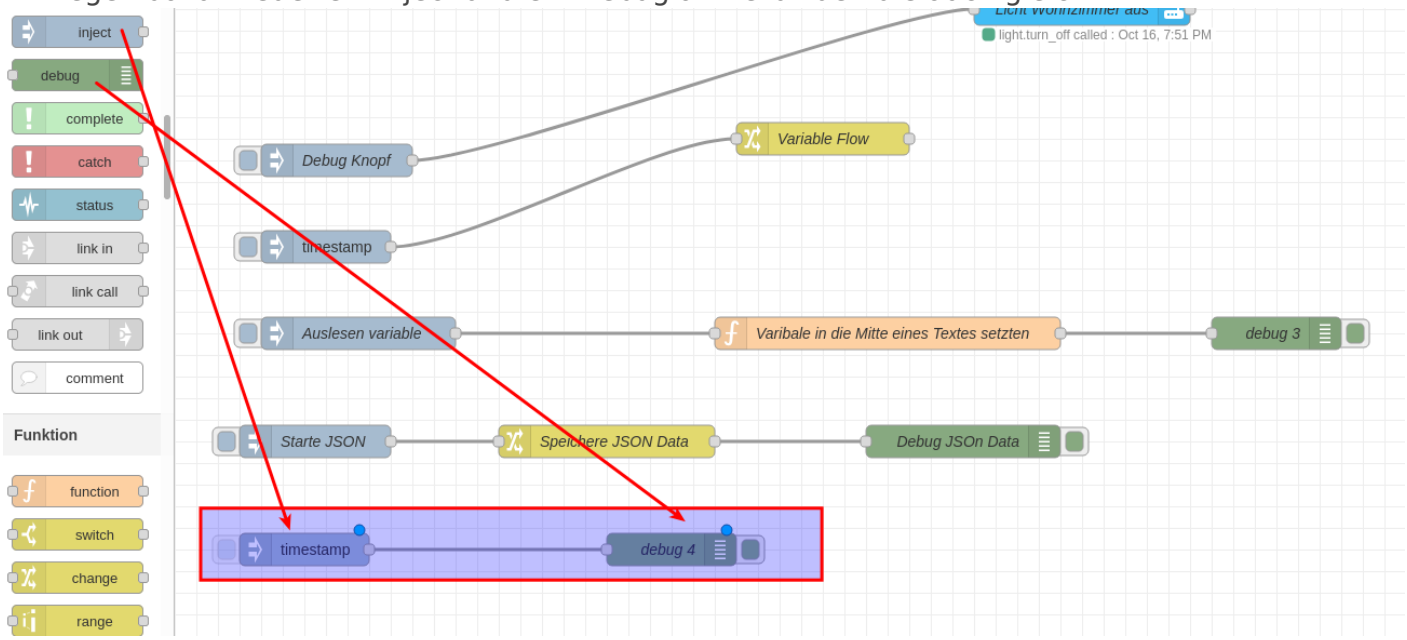


Wenn wir jetzt auf den Injector klicken sehen wir im Debug Fenster unsere json Objekt

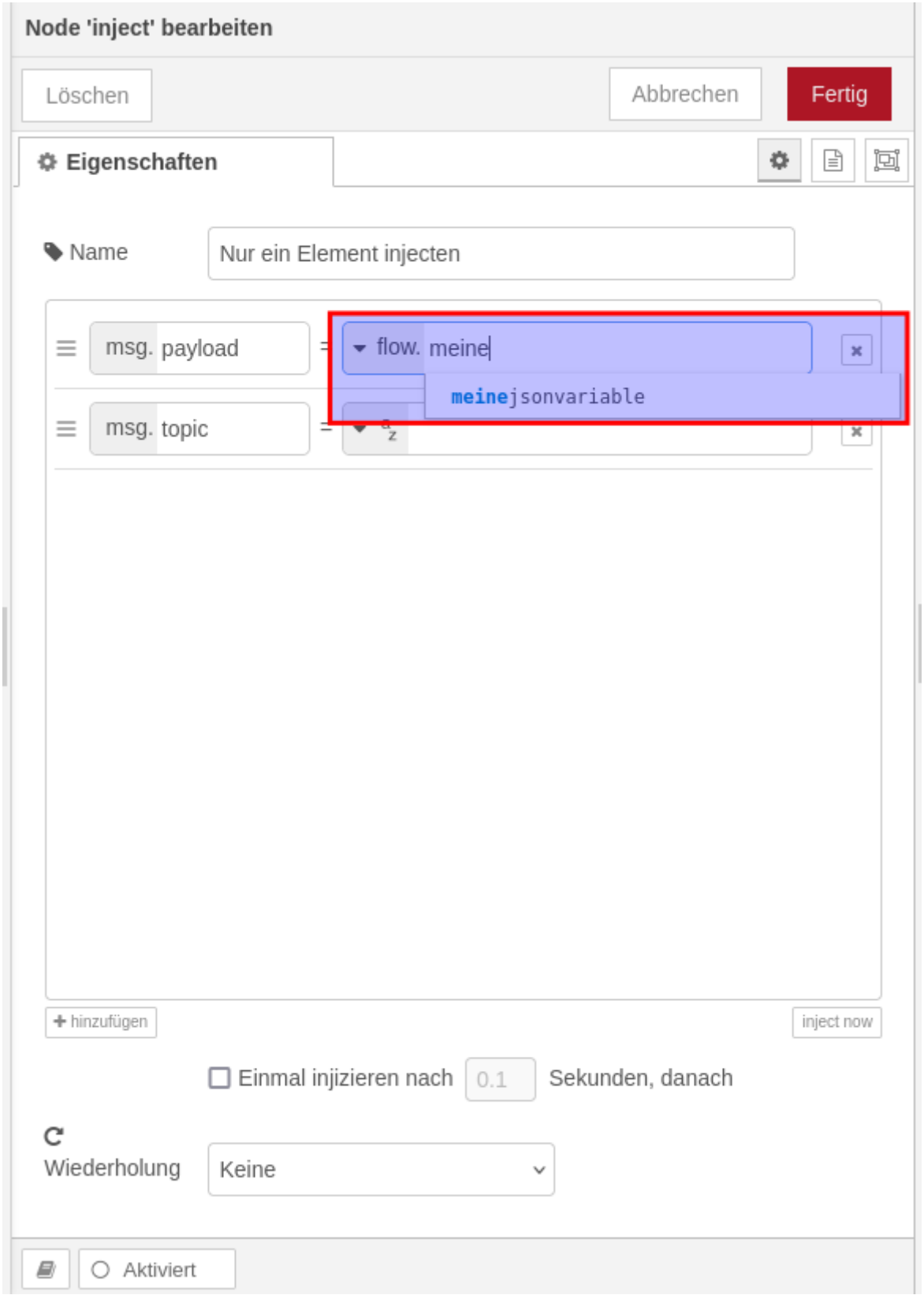


Klicken wir auf den Pfeil gibt ne bessere Übersicht.

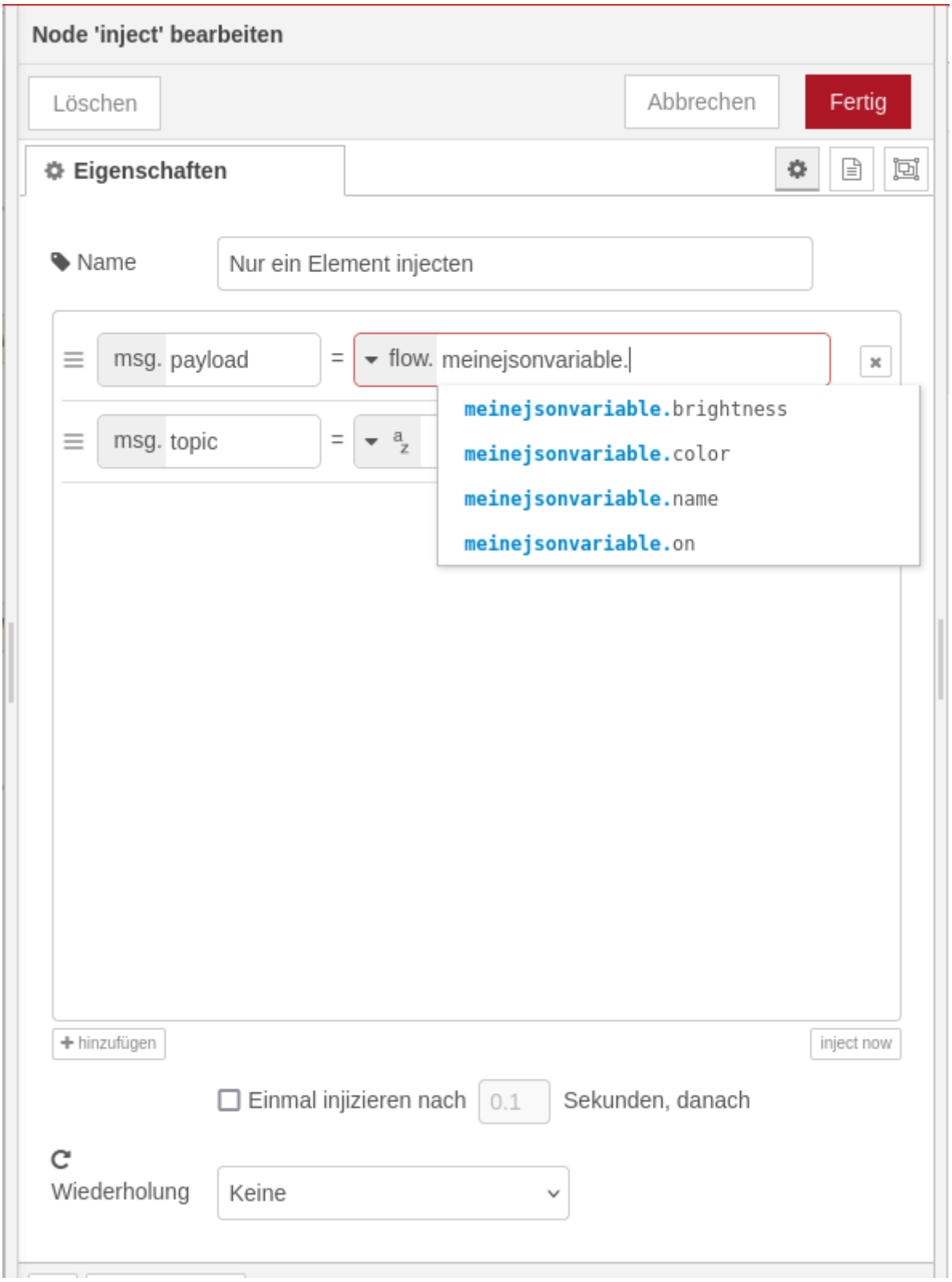
Daten Zugreifen.
Da das ein JSON Object können wir dem "." auf die Objekte zugreifen..
Wir legen dazu wieder ein Inject und ein Debug an Verbinden die auch gleich.



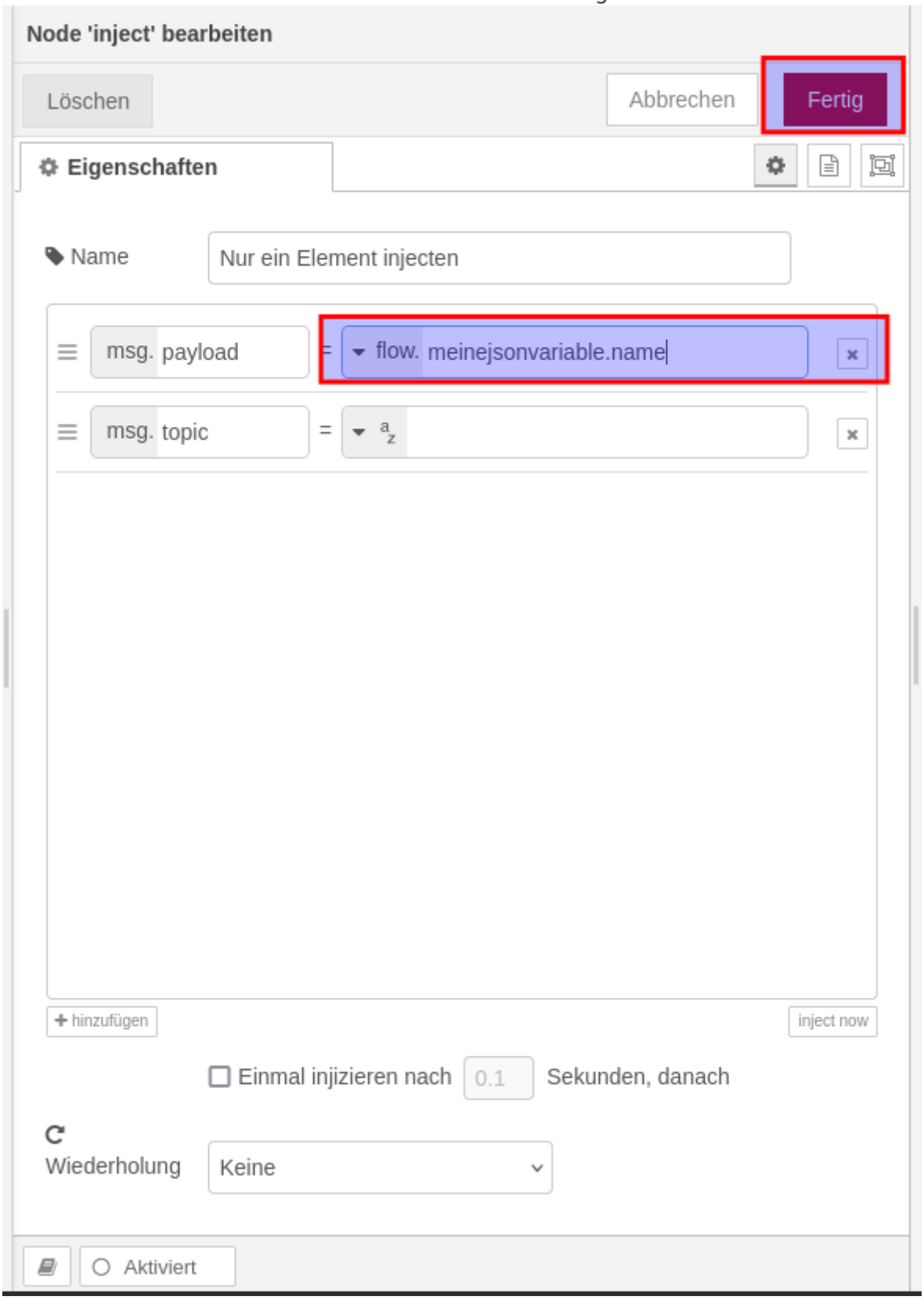
Nun doppelklick auf die Inject node
Dort einen Namen vergeben und dann bei Flow den Variablenamen einegben.



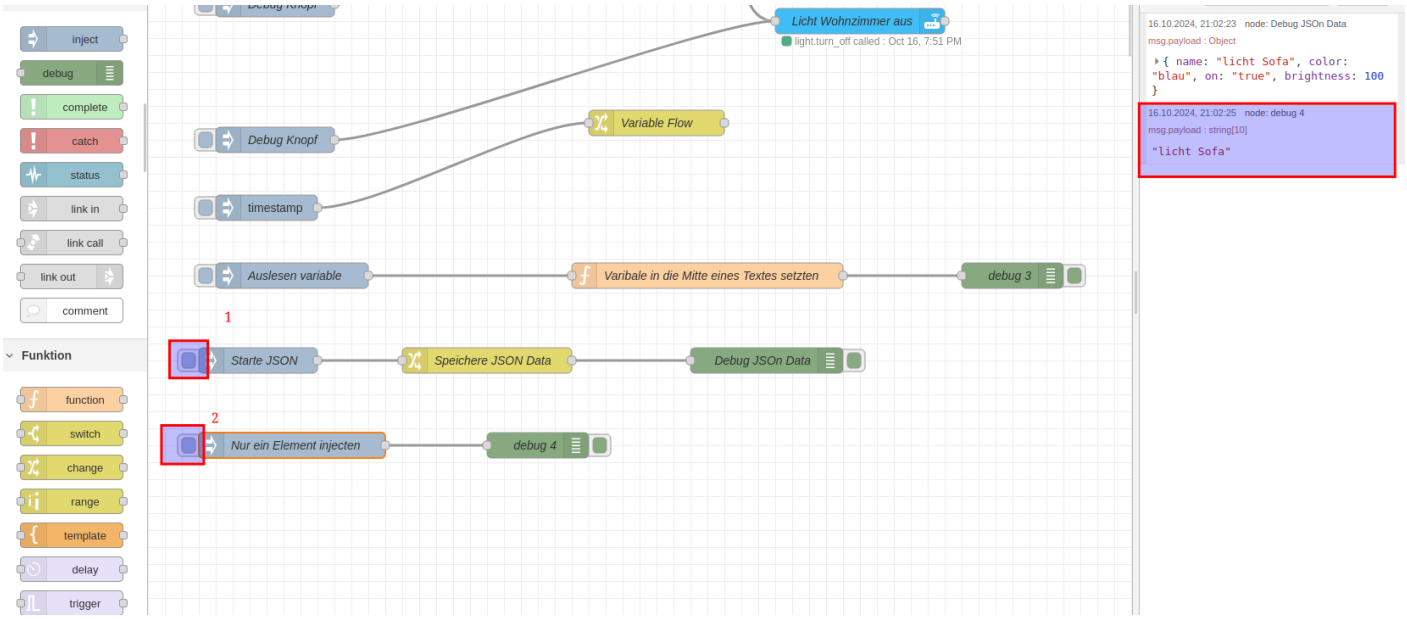
Nun kann hinter den Variablennamen ein punkt angeben werden und wir bekommen eine auflistung der Elemente


Node 'inject' bearbeiten
Löschen
Abbrechen
Fertig
Eigenschaften
Name
Nur ein Element injecten
msg. payload
=
flow. meinejsonvariable.
msg. topic
=
meinejsonvariable.brightness
meinejsonvariable.color
meinejsonvariable.name
meinejsonvariable.on
+ hinzufügen
inject now
Einmal injizieren nach
0.1
Sekunden, danach
Wiederholung
Keine

Wir nehmen hier mal den Namen und klicken auf fertig.



Nun deployen
Dann den ersten inject für die für die json variable erstellen.
Dann den zweiten Inject für das auslesen.
nun sehen wir den Wert der Eigenschaft Name als Payload.
Nämlichen den Namen.



Version #18
Erstellt: 8 September 2023 06:34:42 von Admin
Zuletzt aktualisiert: 16 Oktober 2024 17:03:58 von Admin