ioBroker Back-UP Linux Version

21. Mai

2017

Automatisierte Mehr-Generationen Back-Up und Restore
Vorgehensweise für Linux Installationen

Autor:
Looxer01

Inhalt

Changelog

Datum	Version	Change Log
18.05.2017	BackUp-Restore-Doku	Initiale Version
19.05.2017	BackUp-Restore-Doku_002	Archiv umgestellt auf komprimiert – Danke an Harry423
		MySQL Sicherung hinzugefügt – Danke an Harry423
		Kommentare zu den Shell-Scripten hinzugefügt
		Übersicht der Scripte hinzugefügt.
20.05.2017	BackUp-Restore-Doku_003	Restore von mySQL hinzugefügt
		Script-Übersicht ergänzt - zusätzliche Informationen - Cross Referenzen
		Kapitel „Verzeichnisse“ Anmerkung hinzugefügt
21.05.2017	BackUp-Restore-Doku_004	Textkorrekturen

Einleitung

Die hier beschriebene Vorgehensweise beschreibt eine Alternative zum Standard – Backup von ioBroker. Die Dokumentation richtet sich an erfahrene Anwender.

Voraussetzungen:

Die Prozedur ist ausgerichtet auf ioBroker Installationen unter Linux. Die Linux Distribution sollte keine Rolle spielen. Die hier beschriebene Vorgehensweise muss auf individuelle Belange angepasst werden und kann vermutlich in den meisten Fällen nicht 1:1 übernommen werden.

Die Scripte erstellen die Sicherungen auf dem System auf dem ioBroker installiert ist. Daher sollte genügend Performance vorhanden sein, um die Datenmengen erzeugen zu können. Ein RASPI 3 reicht, allerdings sollte dann ein USB Speicher zur Verfügung stehen. Von einem SD-Speicher wird abgeraten.

Einige Linux-Pakete müssen ggf. nachinstalliert werden z.B. rsync

Die beschriebene Vorgehensweise zeichnet sich aus durch

- Backup und Restore basierend auf Einzeldateien. Damit ist wahlweise ein kompletter restore oder auch ein teilweise restore möglich.
- Die Sicherung erfolgt durch eine Deltasicherung. Im täglichen Betrieb wird die Sicherung daher nur sehr wenig Zeit verwenden
- Die gesicherten Dateien werden in ein einziges Archiv gepackt. Damit kann eine weitere Verteilung wie z.B. auf ein NAS effizient erfolgen
- Die gepackten Dateien werden im Generationen-Prinzip erzeugt. Jedes BackUp hat somit einen eigenen Zeitstempel.
- Die BackUps werden z.B. auf ein NAS kopiert. Somit stehen Kopien zur Verfügung auch wenn die Linux Installation auf der ioBroker sich befindet defekt ist. Das Kopieren kann per push auf das NAS erfolgen oder auch von Windows per pull auf den PC gezogen werden.
- Zur Ergänzung werden alle Archivdateien, die älter als z.B. 14 Tage sind entfernt. Somit kann das BackUp ohne weiteren manuellen Eingriff erfolgen.
- Ein zusätzliches Script erzeugt eine Kopie aller States, so dass bei Verwendung von REDIS auch die States wieder hergestellt werden können.
- Die beschriebene Vorgehensweise benötigt nur minimale downtime von ioBroker. Durch die Kopie aller States ist auch Zero-downtime denkbar.
- Der restore wird ebenfalls beschrieben. Der Restore ist nicht voll automatisiert. Die Einzelschritte werden aber gelistet.

Übersicht Verzeichnisse

Die folgenden Verzeichnisse werden verwendet:

Auf der ioBroker Installation:

- /BackUp/iobroker
  hierhin werden alle Dateien aus dem Verzeichnis /opt/iobroker kopiert (Deltas)
- /BackUp/TARS
  hierhin wird das Verzeichnis /BackUp/iobroker als Archiv-Datei abgelegt
- Das Verzeichnis des NAS muss individuell hinzugefügt werden
- Verzeichnis
- Durch Wiederherstellung von Archiven werden die Verzeichnisse BackUp/TARS/BackUp und /BackUp/TARS/BackUp/iobroker angelegt. Aus diesen kann dann die Rückkopie auf die Originalverzeichnisse erfolgen

Anmerkung:

Empfehlung ist die BackUp und Restore Scripte in /opt zu sichern

Übersicht Sicherungs-Scripte

Name: backup.sh
Type: Shellscript,
Beschreibung: Kopiert iobroker und mysql Daten in das BackUp Verzeichnis. Anschließend werden die kopierten Dateien in ein komprimiertes Archiv gepackt. Jedes Archiv erhält eine Zeitstempel mit Datum und Uhrzeit als Teil des Dateinamens
Aufruf: crontab
Priorität: Dies ist der wichtigste Schritt der Sicherung alle anderen Schritte sind opitional
Siehe Seite: 8

Name: backUPtoUSB
Type: Shellscript
Beschreibung: kopiert die komprimieren Archive auf einen mount (z.B NAS Laufwerk). Dabei werden alle Dateien auf dem mount gelöscht, die sich nicht im Quellverzeichnis befinden
Aufruf: crontab
Priorität: Optional, wenn die Sicherung zusätzlich auf einem mount erfolgen soll
Siehe Seite: 11

Name: delete_Old_Files
Type: shellscript
Beschreibung: Löscht alle Files aus dem BackUP/TARS – Verzeichnis die älter als z.B. 14 Tage sind
Aufruf: crontab
Priorität: Optional, wenn die Sicherung zusätzlich auf einem mount erfolgen soll und alte Dateien automatisch entfernt werden sollen
Siehe Seite: 12

Name: savestates
Type: javascript
Beschreibung: ermittelt alle States der iobroker Objekte und speichert die States in eine Datei. Das ist notwendig bei Nutzung von REDIS, da REDIS aktuelle States im internen Speicher hält.
Aufruf: iobroker-subscription
Priorität: Bei Verwendung von REDIS muss die Sicherung erfolgen
Siehe Seite: 10

Name: BackUpOnRequest
Type: javascript
Beschreibung: damit kann z.B. über VIS ein direktes BackUp angestoßen werden. Dabei werden alle Daten aus VIS, mySQL und States gesichert und in entsprechende Archive gepackt.
Aufruf: iobroker-subscription
Priorität: optional
Siehe Seite: 13

Übersicht Restore-Scripte

Name: uncompress.sh
Type: Shellscript
Beschreibung: entpackt die Archive und speichert die einzelnen Verzeichnisse unterhalb des Verzeichnisses /BackUp/TARS/BackUP
Aufruf: Manuell
Priorität: Bei einem Restore müssen die Dateien entpackt werden
Siehe Seite: 16

Name: restore.sh
Type: Shellscript
Beschreibung: kopiert die Dateien zurück aus den entpackten Verzeichnissen in die Original-Verzeichnisse. Damit wird ioBroker wieder hergestellt.
Aufruf: Manuell
Priorität: Bei einem restore müssen die Verzeichnisse wieder hergestellt werden
Siehe Seite: 17

Name: restoreStates
Type: Javascript
Beschreibung: Stellt die vorher gesicherten States der iobroker Objekte wieder her.
Aufruf: Manuell
Priorität: Bei Verwendung von REDIS müssen die States wieder hergestellt werden
Siehe Seite:18

BackUp

Delta-Kopie und Archiv-Erzeugung

Die Delta-Kopien werden durch den Befehl „rsync“ erzeugt. Sollten aus der Source Dateien gelöscht werden, dann werden diese Dateien auch im Target gelöscht.

Nachdem das BackUp erzeugt worden ist, können die ioBroker Dateien als komprimiertes Archiv gespeichert werden. Vorher wird ioBroker aber noch gestartet und steht somit schneller wieder zur Verfügung.

Quelle für das Archiv ist dabei das gerade erzeugte BackUp in /BackUP/iobroker

Wenn mySQL genutzt wird müssen die entsprechenden Zeilen aktiviert werden

Das unten stehende Shellscript kann z.B. im /opt Ordner hinterlegt werden. Name: backup.sh

Shellscript backup.sh**:**

```
#!/bin/bash

cd /opt/iobroker
iobroker stop

# jetzt wird das iobroker Verzeichnis in das BackUp/iobroker Verzeichnis kopiert
rsync --delete -aLvzh -P /opt/iobroker /BackUp/

# die nächsten Zeilen aktivieren wenn mysql genutzt wird und gesichert werden soll
#cd /BackUp
#/usr/bin/mysqldump --user=root --password=DasPasswort --events --all-databases >
mysql_databases.sql

cd /opt/iobroker
iobroker start

# Erzeugen des Archivs in komprimierter Form
tar -czf /BackUp/TARS/iobroker-20$(date +%y%m%d-%H%M%S).tgz /BackUp/iobroker

# die nächse Zeile aktivieren, wenn mySQL genutzt wird und gesichert werden soll
#/bin/tar -czf /BackUp/TARS/mysql-20$(date +%y%m%d-%H%M%S).tgz
/BackUp/mysql_databases.sql
```

Das Script sollte als cronjob per „crontab –e“ eingeplant werden

Beispiel: 5 0 * * * /opt/backup.sh

Hier wird das Script jeden Tag um 00:05 ausgeführt. Es sollte eine Zeit sein in der ioBroker nicht gestresst ist.

Anmerkung:

Wenn mit REDIS gearbeitet wird, dann kann ggf. auch ohne ioBroker stop gearbeitet werden. Das sollte aber von jedem individuell getestet werden. Bestimmte Situationen könnten zu Inkonsistenzen führen.

Speicherung der States

Wird mit REDIS gearbeitet müssen alle States unbedingt gesichert werden.

Dies wird mit einem **Javascript**, das vor dem BackUp alle States sichert erledigt: hier um 23.50

```
var fs = require('fs');
var now =    new Date(); // store current date and time
var year =   now.getFullYear();
var month =  addZero(now.getMonth()+1).zero2;
var day =    addZero(now.getDate()).zero2;
var Thour =  addZero(now.getHours()).zero2;
var Tmin =   addZero(now.getMinutes()).zero2;
var Tsec =   addZero(now.getSeconds()).zero2;
var logdate = day + '.' + month + '.' + year;
var logtime = Thour + ':' + Tmin + ':' + Tsec;
 datei = "/BackUp/TARS/states";

// Schedule für die Updates
schedule("50 23 * * *",  function () {

var cacheSelectorState = $('state[state.id=javascript.0.*]');
cacheSelectorState.each(function (id, i) {
   var val = getState(id).val;
   var zk = "setState('"+id+"', "+val+");\n";
   if(typeof val === "string") zk = "setState('"+id+"', '"+val+"');\n";
   fs.appendFileSync(datei + logdate + "-"+ logtime + ".txt", zk);
});
   log('States Saved',"info");
 });

// ---------------------------------------------------------------------------------
// Funktion zur Erzeugung von 2 oder 3 führenden Nullen für das Datum Format
//---------------------------------------------------------------------------------------------------
function addZero(i) {
   if (i < 10) {
      j = "00" + i;
      i = "0" + i;
   }
   if (i > 9 && i < 100) {
      j = "0" + i;
   }
   return {
      'zero2' : i,
      'zero3' : j
   };
} // Ende Funktion
```

Kopie auf das NAS

Das mounten des NAS kann je nach Installation anders aussehen. Der mount in diesem Beispiel wird auf ein FritzBox NAS gemacht. Voraussetzung ist, dass auf dem NAS ein User mit Passwort und entsprechender Laufwerksberechtigung angelegt ist.

Bei der Fritzbox ist es wichtig den kompletten Pfad anzugeben wie er auch als Dateiordner in der Fritzbox geführt wird. Hier ist es „Intenso-USB3-0device-01"

Die Konfigurationsdaten zum NAS befinden sich in der Fritzbox unter "Heimnetz" – „Speicher(NAS)".

Das Shellscript erzeugt eine Deltakopie auf dem NAS. Dateien, die in der Source nicht enthalten sind werden auf dem NAS gelöscht.

Shellscript: backUPtoUSB.sh

```
#!/bin/bash

#zunächst wird das NAS gemounted
mount -t cifs -o
username=MyNASUser,password=MyNASPassword,file_mode=0777,dir_mode=0777,uid=1000,gi
d=1000,sec=ntlmv2 //192.168.0.1/fritz.nas/Intenso-USB3-0device-01/iobroker
/mnt/fritzUSB

# jetzt wird alles vom BackUp auf das NAS kopiert. evt vorhandene andere Dateien
werden vom NAS geloescht
rsync --delete -aLvzhP /BackUp/TARS /mnt/fritzUSB

# Wenn ein Unmount gemacht werden soll, dann die nächste Zeile aktivieren
#umount /mnt/fritzUSB
```

In diesem Beispiel wird auf unmount verzichtet

Das Script sollte als cronjob per „crontabe –e" eingeplant werden

Beispiel: 45 1 * * * /opt/backUPtoUSB.sh

Hier wird das Script jeden Tag um 01:45 ausgeführt.

Alte Archiv-Dateien löschen

Damit alte Archiv-Dateien nicht zu einem Überlauf des Speichers führen, können alte Dateien, die älter sind als x Tage automatisch per shellscript gelöscht werden. In diesem Beispiel sind es 14 Tage.

Das untenstehende Script löscht sowohl alte Archive als auch alte State und mysql - Dateien, die durch die Sicherung der States und mysql entstanden sind

Shellscript: Delete_Old_Files.sh

```
#die folgende Zeile loescht alle Dateien im TARS Verzeichnis, die aelter als 14
Tage sind und mit "states" beginnen
find /BackUp/TARS -name 'states*' -mtime +14 -exec rm -rf {} \;

#die folgende Zeile loescht alle Dateien im TARS Verzeichnis, die aelter als 14
Tage sind und mit "iobroker" beginnen
find /BackUp/TARS -name 'iobroker-*' -mtime +14 -exec rm -rf {} \;

#die folgende Zeile loescht alle Dateien im TARS Verzeichnis, die aelter als 14
Tage sind und mit "mysql" beginnen
find /BackUp/TARS -name 'mysql*' -mtime +14 -exec rm -rf {} \;
```

Beim nächsten NAS Sync Lauf warden diese Dateien auf vom NAS geloescht.

Das Script sollte als cronjob per „crontabe –e“ eingeplant werden

Beispiel: 45 2 * * * /opt/Delete_Old_Files.sh

Hier wird das Script jeden Tag um 02:45 ausgeführt.

BackUp anstoßen aus VIS

Das nachfolgende Script erzeugt einen State. Wenn dieser auf true gesetzt wird, dann werden alle States gesichert (nur notwendig für REDIS) und das backup.sh Script wird aufgerufen. Damit fährt auch ioBroker (falls gewünscht) runter und es dauert eine Weile bis VIS wieder oben ist.

```
createState('BackUp-Flag', false);                          // BackUp Flag
var BackUpVar   =  "BackUp-Flag";                    // BackupFlag
// main
// ------------------------------------------------------------------------------
on( {id:BackUpVar, val: true }, function(obj) {                 // Event:

// if (getState(BackUpVar).val === true )  {                      setState(BackUpVar,false) ;
log("EVENT Backup gestartet " ,"info");
StatesSave();
BackUp();
// } // ende if BackUpFlag Check

 }); // ende on id

 function BackUp() {
   exec('/opt/backup.sh', function(err, stdout, stderr) {
      if (err) {
         log(err);
         return;
      }
      stdout = stdout.replace(/[\D]+/, "");   // alle Zeichen vor der ersten Ziffer entfernen
      stdout = stdout.split(/[\D]+/g);       // alle nicht-Ziffern als Trennzeichen für das Array verwenden (im Block)
      log(stdout);
      // writeDp(stdout);
   });
}

function StatesSave() {
var fs = require('fs');
var now =    new Date(); // store current date and time
var year =   now.getFullYear();
var month =  addZero(now.getMonth()+1).zero2;
var day =    addZero(now.getDate()).zero2;
var Thour =  addZero(now.getHours()).zero2;
var Tmin =   addZero(now.getMinutes()).zero2;
var Tsec =   addZero(now.getSeconds()).zero2;
var logdate = day + '.' + month + '.' + year;
var logtime = Thour + ':' + Tmin + ':' + Tsec;

datei = "/BackUp/TARS/states";

var cacheSelectorState = $('state[state.id=javascript.0.*]');
cacheSelectorState.each(function (id, i) {
   var val = getState(id).val;
   var zk = "setState('"+id+"', "+val+");\n";
   if(typeof val === "string") zk = "setState('"+id+"', '"+val+"');\n";
   fs.appendFileSync(datei + logdate + "-"+ logtime + ".txt", zk);
});


}
```

```
// ---------------------------------------------------------------------------------
// Funktion zur Erzeugung von 2 oder 3 führenden Nullen für das Datum Format
//---------------------------------------------------------------------------------------------------
function addZero(i) {
   if (i < 10) {
      j = "00" + i;
      i = "0" + i;
   }
   if (i > 9 && i < 100) {
      j = "0" + i;
   }
   return {
      'zero2' : i,
      'zero3' : j
   };
} // Ende Funktion
```

Restore

Der Restore kann durch eine komplette Kopie eines BackUps erfolgen oder auch durch individuellen Austausch von einzelnen Dateien, beispielsweise der Views. Während der Austausch von individuellen Dateien ein rein manueller Vorgang ist, kann der restore durch teilautomatisch durchgeführt werden.

Warum teilautomatisch ?

- Beispielsweise kann entschieden werden, ob das alte ioBroker Verzeichnis behalten werden soll. In diesem Fall kann ganz einfach das ioBroker Verzeichnis umbenannt werden. Der Anschließende Restore erzeugt dann ein neues ioBroker Verzeichnis mit den Dateien aus dem BackUp (der rename ist empfohlen – sollte aber später gelöscht werden)
- Es muss entschieden werden, welches Backup verwendet werden soll. Hierzu muss im Shellscript der entsprechende Dateiname eingegeben werden.

Schritte für den Restore:

1. Rename das Verzeichnis /opt/ioBroker z.B. in /opt/iobroker_old
2. Löschen des BackUp Verzeichnisses /BackUp/iobroker
3. Uncompress die gewünschte Archivdatei
4. Kopie der entpackten Dateien in das Verzeichnis /opt/iobroker
5. Wiederherstellung der States

Umbenennen des ioBroker Verzeichnisses

Zunächst ioBroker stoppen:
cd/opt/iobroker
iobroker stop
Das Umbenennen kann jetzt durch das SSH Tool gemacht werden oder manuell auf der Kommandozeile mit: mv /opt/iobroker /opt/iobroker_OLD

Wenn iobroker nach dem Restore einwandfrei läuft kann das iobroker_OLD Verzeichnis wieder gelöscht werden: rm -rf /opt/iobroker_OLD

Löschen des BackUp Verzeichnisses

Sicherheitshalber sollte das BackUp Verzeichnis gelöscht werden. Damit wird sichergestellt, dass keine ungewünschten Dateien wieder hergestellt werden: rm -rf /BackUp/iobroker

Uncompress der Archivdateien

Die Archivdateien werden durch ein einfaches entpacken in das ursprüngliche Verzeichnis zurückgeschrieben. Der Dateiname muss angepasst werden also mit dem gewünschen BackUp Datum versehen werden.

Shellscript: uncompress.sh

```
# uncompress der Datei in der Ursprungsfolder /BackUp/iobroker
# Die Dateinamen muessen angepasst werden entsprechend Datum und Uhrzeit
# Entpackt wird in Folder /BackUp/TARS/BackUp/iobroker

#Entpacken des ioBroker Verzeichnis
tar -xvf /BackUp/TARS/iobroker-20170519-114035.tgz
```

Achtung: Dateinamen sind exemplarisch

Anmerkung: Die entpackten Dateien finden sich im Verzeichnis
`/BackUp/TARS/BackUp/iobroker wieder. Das Ursprungs-Verzeichns /BackUp/iobroker wird nicht geaendert`

Wiederherstellung der ioBroker Dateien

Nachdem die Archivdateien entpackt worden sind, kann jetzt die Kopie zurück zu iobroker erfolgen.

Shellscript : restore.sh

```
#!/bin/bash
cd /opt/iobroker
iobroker stop

#Wiederherstellen des iobroker Verzeichnisses
rsync  -aLvzh /BackUp/BackUp/iobroker /opt/

#Bei Verwendung von mySQL bitte die folgende Zeile aktivieren - Wiederherstellen
der mySQL Datenbank
#tar -xzOf your_db_dump.tgz | mysql --user=root --password=DeinPasswort

.iobroker start
```

Das Script kann auf der Kommandooberfläche aufgerufen werden z.B. durch /opt/restore.sh

Wiederherstellung der States

Zur Wiederherstellung der States (falls gewünscht) muss ioBroker zunächst gestartet werden. Es ist empfehlenswert alle Instanzen, die State Updates vornehmen zu stoppen mit Ausnahme von JavaScript.

Dann muss ein neues Script angelegt werden und die gesicherte States-Datei in das Script eingefügt. Nachdem das Script ausgeführt ist sollte das Script sofort gelöscht werden.

Achtung: bei einem langsamen Rechner empfiehlt sich die Aufteilung z.B. in 500 Zeilen Schritten

Einzelschritte:

1. Starte ioBroker
2. Stoppe alle Instanzen, die States fortschreiben mit Ausnahme von Javascript
3. Erstelle ein neues JavaScript Programm und Paste die gesicherten States der richtigen States Sicherung in das Programm. (evtl. in 500 er Blöcken)
4. Programm ausführen
5. Nach Erfolgreicher Ausführung : Programm löschen
6. Reaktivierung der Instanzen