# Visbo ReST Server and UI installation on AWS

Hier finden Sie eine Beschreibung der ToDos zum Einrichten des Visbo ReST-Servers einschließlich UI und Dokumentation in einer AWS-Umgebung.
Wenn Sie nur nach einer Upgrade-Anleitung suchen, besuchen Sie bitte diese Seite: Visbo ReST Server and UI Update on AWS

Ausgangspunkt ist eine AWS Management Console, in der Sie eine Maschine und ein Betriebssystem auswählen können. Die beschriebenen Schritte umfassen die Installation und das Update/Upgrade von Basissoftware wie NGINX, Node, GIT und PM2.
Aber auch die Installation der Visbo Software und die Konfiguration der Komponenten, so dass am Ende des Prozesses ein laufender Server mit SSL steht.

Um den Server vor Angriffen zu schützen, müssen wir UFW nicht einrichten, da Amazon die Möglichkeit bietet, die eingehenden Anrufe an bestimmte Ports in der Management Console zu verwalten. Auch die Verarbeitung des HTTPS-Datenverkehrs erfolgt auf dem Load Balancer anstelle des lokalen nginx. Bedeutet keine Zertifikatsinstallation auf EC2 und ein anderes Setup für nginx.

Da wir mit AWS flexibler sind, führen wir die Standardeinrichtung mit mindestens 2 Anwendungsservern im Vergleich zu einem App-Server in der 1&1-Umgebung durch. Dies bringt zusätzliche Komplexität mit sich, da wir Loadbalancing für die mehreren Server, ein gemeinsames Speichersystem für die Protokolldateien und den Redis-Cache benötigen, der von allen Anwendungsservern verwendet wird.
Später müssen wir die Möglichkeiten des dynamischen Hinzufügens und Entfernens von Ressourcen bewerten, damit das System bei hoher Last und danach wachsen und schrumpfen kann.

Für diese Konfiguration habe ich mich gegen die Verwendung von Containern und Container-Management wie Kubernetis Software entschieden, da es meiner Meinung nach nicht für den ReST-Server als einzelne App passt. Serverkonfiguration.

### Schritt-für-Schritt-Anleitung

Dies sind die erforderlichen Schritte:

1.

2.
    a. Einrichten einer VPC in einem der Standard-IP-Bereiche 172.16.0.0/12 - 172.31.255.255/12 (um später Peering einrichten zu können)
    b.
        i. Automatische Zuweisung öffentlicher IP-Adressen
    c.
        i. Route zur VPC hinzufügen
    d. DNS-Auflösung und DNS-Hostnamen für VPC aktivieren
    e.
        i. Standard (eingehendes HTTP/HTTPS von überall, ausgehend)
        ii. ReST (eingehendes HTTP/HTTPS von lokalen IPs, NFS von lokaler IP, ausgehend alle)
        iii. SSH (eingehendes SSH von bestimmten IPs, ausgehende keine)
    f.
        i. Peering-Verbindung in Atlas erstellen
        ii. Peering-Verbindung in AWS genehmigen
        iii. Route in VPC hinzufügen
3.
    a. Befolgen Sie diese Schritte, um es von Grund auf neu zu installieren
    b. Gehen Sie folgendermaßen vor, um es aus einem installierten Image zu erstellen

**Multi Server mit Loadbalancer ohne Auto Scaling einrichten**

Nachdem das obige Setup durchgeführt wurde und alles funktioniert, besteht der nächste Schritt darin, Multiserverfunktionen einzurichten. Der erste Schritt besteht darin, mehrere ReST-Server zu haben, die die Anfragen bedienen, später könnte dies dynamisch konfiguriert werden, so dass Amazon je nach Lastbedingungen ein neues Image startet und an den Load Balancer anhängt oder ein bestehendes stoppt.

1.
    a. Setup-Zielgruppe, die die ec2-Instanzen und die Zustandsprüfungen für den Load Balancer sammelt.
    b. Nur HTTP & HTTPS zulassen
    c. Zertifikat von ACM zuweisen
    d.
        i. Verwenden Sie HTTP /api/status

    ii. Konfigurieren Sie die Häufigkeit und die Zeit, wie lange es nach erfolgreicher Antwort dauert, bevor sie fehlerfrei wird
  e. Generieren eines Images aus EC2 und Starten einer zweiten Instance
  f. Zuweisen der zweiten Instanz und Überprüfen, ob die Last geteilt ist
2.
  a. einerseits für den Umgang mit IP-Adressen
  b. UI und API Doc als Webseiten bereitstellen
3. Mongo Secure Network Setup
   mit dynamischen EC2-Instances ist es nicht sinnvoll, IP-Einschränkungen mit Atlas MongoDB zu verwenden. Stattdessen bieten Atlas und AWS ein Peering von Netzwerken an, sodass dem Netzwerkzugriff von Visbo EC2-Instances auf die Atlas Mongo AWS-Konfiguration vertraut wurde. Es gibt einen Kommentar, dass dies nach 7 Tagen abläuft, aber ich nehme an, dass sich dies nur auf das nicht gebundene Peering bezieht. Wenn dies konfiguriert ist, kann die neue EC2-Instance ohne zusätzliche Konfiguration auf die Mongo-DB zugreifen.
4.
  a. Erstellen Sie die Startvorlage mit allen voreingestellten Dingen außer den Namen für EC2 & Volume
  b. Erstellen Sie eine EC2 aus dieser Vorlage und weisen Sie den Namen der EC2 und dem Volume zu.
  c. Nachdem die EC2-Instance angegeben wurde, weisen Sie sie der Load Balancer-Zielgruppe zu
5.
  a. Konfigurieren Sie entweder Ihre eigene Instance oder verwenden Sie die AWS ElastiCache-Konfigurations
     https://eu-central-1.console.aws.amazon.com/elasticache/home?region=eu-central-1
  b. Kopieren Sie den Endpunkt-Hostnamen und den Port und konfigurieren Sie ihn in Mongo als SystemConfig-Einstellung

**Multi Server mit Loadbalancer inklusive Auto Scaling einrichten**

Nachdem das obige Setup abgeschlossen wurde und alles funktioniert, besteht der nächste Schritt darin, die Konfiguration so zu ändern, dass wir je nach Last eine dynamische Anzahl von Servern erhalten, um die ReST-Anforderungen zu bedienen.
Wie wir die obigen Schritte ausgeführt haben, haben wir eine konfigurierte und funktionierende Instanz des ReST-Servers einschließlich einer Startvorlage. Damit die automatische Skalierung funktioniert, führen wir die folgenden Schritte aus:

1. Erstellen Sie ein Image Ihrer ausgeführten Instance, das später als AMI dient. Unsere Namenskonvention ist es, es "2019-08-21_VisboStagAMI zu nennen
2. 
    a. Verwenden der neu vorbereiteten AMI-ID
    b. Entfernen Sie die snapshotID für das Volume (dies ist kein Problem beim manuellen Starten, aber bei AutoScaling schlägt dies fehl)
    c. Nach dem Entfernen der snapshotID müssen Sie die Verschlüsselung überprüfen, um mit einem Chiffrierschlüssel auf Ja gesetzt zu werden. Wenn Sie die snapshotID löschen, wird sie auf unverschlüsselt gesetzt.
    d. Konfigurieren Sie die Tags Area & Name und weisen Sie sie sowohl der Instanz als auch dem Volume zu
3. 
    a. Wählen Sie die Option aus, um die Startvorlage zu verwenden
    b. Wählen Sie die gewünschte Startvorlage aus und verwenden Sie "Neueste Version" für die ausgewählte Version
    c. Konfigurieren Sie Mindestgröße 1, gewünscht 1 Max 1
    d. Zuweisen der Auto Scaling-Gruppe zur Load Balancer-Zielgruppe
4. 
    a. Es muss sich im Status befinden, der in der Instanzliste ausgeführt wird
    b. es muss ein Mitglied der Zielgruppe sein
    c. Es muss nach einer Anfangsphase gesund sein
5. 
    a. Ändern der Anzahl der Instanzen (min, desired, max)
    b. Hinzufügen von Skalierungsregeln zur Gruppe
    CPU-Last als Indikator zum Starten/Stoppen neuer Instanzen verwenden
    c. Hinzufügen von Benachrichtigungseinstellungen zu den Aktionen Erstellen / Beenden von Auto Scaling
    d. Laden Sie das System und überprüfen Sie, ob neue Instanzen wie gewünscht gestartet und gestoppt wurden
6. 
    a. Die CPU-Last und die Anzahl der Anforderungen können pro Auto Scaling-Gruppe erfasst und überwacht werden

1. Bestellen Sie eine Maschine im jeweiligen Rechenzentrum, für Visbo nutzen wir Frankfurt als bevorzugtes DataCenter.
2. Wählen Sie eines der VisboAMIs aus, die Sie als Vorlage verwenden möchten
3. 
    a. Wählen Sie die VPC aus.
    b. Wählen Sie die 3 Sicherheitsgruppen
    c. Erstellen von Tags für Name und Bereich
    d. Überprüfen Sie auf der letzten Seite, ob die Instanz eine öffentliche IP/einen öffentlichen Namen erhält
4. Erstellen oder Auswählen eines Schlüsselpaars, um über SSH eine Verbindung zum Server herstellen zu können Speichern Sie die SSH-Schlüssel
   auf dem lokalen Computer an einem sicheren Ort
   Melden Sie sich am Computer an, d. h.: ssh -i /Users/markus/Documents/AWS/AWSVisboDevFra.pemubuntu@ec2-3-122-236-148.eu-central-1.compute.amazonaws.com
5. 
    a. 
        i. cd $HOME/GitHub
        ii. mv visbo-server-rest delete-visbo-server-rest; m v visbo-server-uidelete-visbo-server-ui
        iii. git config --global user.name stashReader
        iv. git config --global user.email stash@visbo.de
        v. git init
        vi. SEIEN SIE VORSICHTIG MIT DEN MINUSZEICHEN, IRGENDWIE WERDEN SIE BEIM KOPIEREN UND EINFÜGEN DURCH ANDERE ZEICHEN ERSETZT
        vii. git clonehttps://stashReader@github.com/visbo-open-source/visbo-server-rest.git--single-branch--branch XXXXX
        viii. npm install // Wechseln Sie zu den neuen Verzeichnissen und führen Sie den Befehl install aus
        ix. cd $HOME/GitHub/visbo-server-rest; Bearbeiten der .env-Datei Kopieren Sie die Vorlage aus dem delete-visbo-server-rest
        x. git config credential.helper store // innerhalb der Ordner aufzurufen

xi. cd $HOME/GitHub/visbo-rest-ui; ng update @angular/cli // update config file delete .angular-cli.json create angular.json In diesem Fall war die lokale CLI 1.7 und

6.
   a. Datei /etc/nginx/sites-enabled/dev.visbo.net
   b. Abschnitt: Speicherort /api
   c. Reihe hinzufügen: client_max_body_size 5m;
7.
   a. Erstellen eines EFS-Dateisystems mit einem Sicherheitsprofil in AWS Management
   b. Verwenden Sie die spezifische Sicherheitsgruppe VisboGroupReST in VPC und weisen Sie sie dem EFS FS und der EC2-Instance zu.
   c. Achten Sie darauf, dass das Reittier nach jedem Neustart ausgeführt wird. Übernehmen Sie "$HOME/bin/aws-automount.sh" für das richtige EFS-Volume
   d.
      i. /var/log/visbo
      ii. $HOME/.pm2/logs
8.
   a. REDIS-Server in AWS Elasticache / Redis einrichten
      i. Zuweisen zur VPC
      ii. Subnetzgruppe erstellen
      iii. Wählen Sie t2.micro REDIS Server
      iv. 0 Replikationen im Moment
      v. Kein Speichern auf Festplatte
      vi. Zuweisen der ReST-Sicherheitsgruppe zum Redis-Cluster
      vii. Hinzufügen der Berechtigung zur ReST-Sicherheitsgruppe, um die Kommunikation zwischen ReST und Redis
      Custom TCP Rule zuzulassen, Port 6379 nur auf Rest-Sicherheitsgruppe beschränken
9.
   a.
      i. Umgebung für ReST Server Configuration dist/env übernehmen.js
      ii. Update-UI
   b.
      i.
         1. DB-Verbindungszeichenfolge für ReST-Server

2. CORS-Einstellung

ii. update DB Sysconfig "UI Url"

iii. Update-Rest-Kraft

10.

a. Erstellen eines neuen Load Balancers (Anwendungsmodus)

b. Weisen Sie die richtige VPC zu und verwenden Sie die Standardsicherheitsgruppe für den Loadbalancer (eingehenden HTTP/HTTPS-Datenverkehr zulassen).

c. Wählen Sie die 2017 TLS 1.2-Richtlinie für den Loadbalancer aus.

d. Übernehmen Sie die Zustandsprüfung in /api/status

e. Neue Zielgruppe anlegen

f. Zuweisen Ihrer Instance zu dieser Zielgruppe

g.

i. HTTP ist so konfiguriert, dass es zu HTTPS umgeleitet wird

ii. HTTPS leitet an HTTP der Zielgruppe weiter

11.

a. Erfassen des DNS-Namens des Load Balancers für die Konfiguration

b. in IONOS eine neue Subdomain mit dem neuen Namen einrichten

c. Entfernen Sie alle Eigenschaften und erstellen Sie einen CNAME-Eintrag mit der Umleitung zum DNS-Namen des Loadbalancers

d. Halten Sie die TTL am Anfang kurz (5 Minuten). Dies erleichtert den Wechsel bei Bedarf

12.

1. Bestellen Sie eine Maschine im jeweiligen Rechenzentrum, für Visbo nutzen wir Frankfurt als bevorzugtes DataCenter.
2. Wählen Sie Ubuntu 22.04 oder höher als Betriebssystem
3. Erstellen oder Auswählen eines Schlüsselpaars, um über SSH eine Verbindung zum Server herstellen zu können Speichern Sie die SSH-Schlüssel
   auf dem lokalen Computer an einem sicheren Ort
   Melden Sie sich am Computer an, d. h.: ssh -i
   /Users/markus/Documents/AWS/AWSVisboDevFra.pemubuntu@ec2-3-122-236-148.eu-central-1.compute.amazonaws.com

4. Eine Liste von Befehlen befindet sich in GIT visbo-server-rest/setup/init-aws-instance , aber führen Sie sie nicht aus, da das Skript nicht funktioniert und keine Vorbedingungen überprüft
5. 
    a. Passwort des root-Benutzers ändern
    b. Ubuntu-Benutzer wird für ssh verwendet, aber nur mit Schlüsselpaar
6. Verwenden Sie den Ubuntu-Benutzer anstelle von Visbo für die Anwendung, es hat sudo-Rechte
7. 
    a. /usr/lib/update-notifier/apt-check -p// Holen Sie sich die Update-Liste
    b. sudo apt-get update // Holen Sie sich die Liste der Updates
    c. sudo apt-get upgrade -y // aktualisiert strikt die aktuellen Pakete
    d. sudo apt-get dist-upgrade // installiert Updates (neue)
    e. lsb_release -a // Holen Sie sich die Ubuntu-Release-Informationen
    f. sudo apt-get install nginx // um ein neues Paket von nginx zu installieren
8. 
    a. /usr/bin/landscape-sysinfo
    b. sudo cat /var/lib/update-notifier/updates-available
9. 
    a. sudo apt-get install -y nodejs
    b. sudo apt installieren npm
    c. sudo npm install -g n // install Modul n, das für das Upgrade verwendet wird
    d. sudo n stable // Holen Sie sich die neueste stabile Version
    e. sudo npm install -g @angular/cli // Installieren Sie die Angular-Befehlszeilenschnittstelle, -g für global
    f. sudo npm install apidoc -g// Installieren Sie das APIDOC-Tool, das API-Dokumente von der Quelle ins Web konvertiert
10. 
    a. 
        i. / Root-Gerät
        Kann klein gehalten werden, da es nicht viel wächst
        ii. /usr
        kann auch klein gehalten werden IT speichert zusätzliche Software

iii. /var

The main area for this filesytem is logging (System, UFW, NGINX, Visbo, ...) and it keeps also the Web Site information (UI and API Doc)

Especially the Logging could grow quite extensively, so the default of 5 GB is too small, even for Staging. Configure at least 30 GB.

iv. /home

This is the device for the users folder. Here the software and all its components were stored. And with the default config also the pm2 logs. This is a duplication of the Visbo Logs. Take care that the pm2 logs gets deleted on a regular base.

11. Configure your certificate in Certificate Management (ACM)
upload private key, public key and intermediate key
12.

a.

i. Default Group (Allow Inbound HTTP & HTTPS for Loadbalancer)
ii. ReST Group (Allow HTTP, NFS and REDIS from the REDIS Group)
iii. SSH (Allow SSH to the instance from defined IP addresses)

13.

a.

i. git config --global user.name stashreader
ii. git config --global user.email <email adress of admin>
iii. git init
iv. git status // shows the status of the repository including actual branch and local diffs
v. git remote show origin // shows all the branches on the remote server
vi. git remote set-head origin staging // set the active branch to staging
vii. git checkout — file // removes local modification without touching central repositor
viii. git config credential.helper store // store the credentials during next authentication and reuse it later
ix. git pull // gets the newest version of the branch by getting all files
x. git fetch // gets something??? from stash

b.

i. mkdir $HOME/GitHub; cd $HOME/GitHub
ii. git config --global user.name stashreader

iii. git config --global user.email <email adress of admin>
iv. git init
v. BE CAREFUL with the minus signs, somehow they get replaced by other charactsrs during copy & paste retype them manually
vi. git clone https://stashReader@github.com/visbo-open-source/visbo-server-rest.git --branch staging --single-branch
vii. npm install // go to the new directories and run the install command
viii.
1. https://stackoverflow.com/questions/43931986/how-to-upgrade-angular-cli-to-the-latest-version
ix. cd $HOME/GitHub/visbo-rest-ui; ng update @angular/cli // update config file delete .angular-cli.json create angular.json in this case the local cli was 1.7 and

14.
a. https://tecadmin.net/setup-nginx-as-frontend-server-for-nodejs/
b. mkdir $HOME/MyApp
c. Create File myapp.js with test server
d. node myapp.js & // run it locally

15.
a.
i. /etc/nginx // the config directory for nginx
ii. /etc/nginx/nginx.conf // the main configuration, typically not necessary to change
iii. /etc/nginx/sites-available // for each site a separate file that contains the configuration
iv. /etc/nginx/sites-enabled // contains symbolic links to sites-available and only these sites are active
v. /var/log/nginx // HTTP LOGs and Error Logs
vi. use the config file from the GIT ReST Server Package (visbo-server-rest/setup) to configure your NGINX, we do not use SSL on AWS, as we terminate it already on the load balancer.
b.
i. sudo systemctl stop nginx
ii. sudo systemctl start nginx
iii. sudo systemctl restart nginx
iv. sudo systemctl enable nginx // to configure nginx to start after reboot
v. sudo systemctl reload nginx

c.
    i. sudo apt-get install software-properties-common
    ii. sudo add-apt-repository ppa:nginx/stable // declare nginx as package that will be checked during upgrade
    iii. sudo apt-get install nginx // standalone installation without upgrade check
    iv. sudo systemctl start nginx
    v. sudo systemctl enable nginx
    vi. Browse to the IP address or to the CNAME to see the NGINX Page, proves that the site is available from outside
d.
    i.
        1. sites-available
        2. we don't need Certificates here, as we terminate SSL on the load balancer

16.
a. Create and remove swap files in Ubuntu
b. sudo swapon -s // to check if swap file is configured
c. sudo fallocate -l 1.5G /var/swap.1 // to create a 1GB Swap file
d. sudo mkswap /var/swap.1 // make it a swap file
e. sudo swapon /var/swap.1 // mount it as swap file
f. sudo swapon -s // show that it is available
g. ADD: /var/swap.1 swap swap defaults 0 0 // to /etc/fstab
h. sudo swapoff -a // disable all swap files

17.

a. Create in AWS Management an EFS Filesystem with a Security Profile
b. Use the Security Group VisboGroupReST in VPC and assign it to the EFS FS and to the EC2 instance
c. sudo mkdir /var/centrallog // mount location
d. install the aws-mounting utilities (see: init-aws-setup)
e. sudo mount -t nfs4 -o nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,noresvport fs-89b282d0.efs.eu-central-1.amazonaws.com:/ /var/centrallog // for one time mounting
f. take care that the mount will be executed after each reboot. An entry inside /etc/fstab did not work, might be because some permissions or network groups were not assigned yet.

We create a service config that gets installed and executed during boot time.
To initiate the service see: https://wiki.ubuntuusers.de/systemd/Service_Units/
The service asks for "multi-user.target" so that the basic network setup has finished already. The service executes the aws_automount script in $HOME/bin where the EFS parameters like name and options were stored and can be adopted to the specific need.
Adopt "$HOME/bin/aws-automount.sh" for the correct EFS Volume.
sudo systemctl enable $HOME/GitHub/visbo-server-rest/setup/aws-visbo.service // creates the service and creates symbolic links in /etc/systemd/system to the location of the aws-visbo.service

g.
    i. sudo systemctl status aws-visbo // to get the last few lines of the service output

h.
    i. /var/log/visbo
    ii. $HOME/.pm2/logs

18.
    a. Create a new Load Balancer (Application Mode)
    b. Assign the correct VPC and use the default security Group for the Loadbalancer (allow HTTP/HTTPS incoming traffic)
    c. select the 2017 TLS 1.2 Policy for the Loadbalancer
    d. adopt the health check to /api/status
    e. Create a new Target Group
    f. Assign your instance to this target group
    g.
        i. HTTP is configured to redirect to HTTPS
        ii. HTTPS is forwarding to HTTP of the target group

19.
    a. collect the DNS name of the load balancer for the configuration
    b. setup in IONOS a new subdomain with the new name
    c. remove all properties and create a CNAME entry with the redirect to the DNS name of the loadbalancer
    d. keep the TTL in the beginning short (5 Minutes). This makes it easier to change if necessary

20.
    a. cd $HOME
    b. mkdir bin // create bin directory for upgrade rest&ui commands

c. cp update-rest and update-ui files from setup folder to bin
d. chmod ug+rwx bin/update* // add execute permission
e.
    i. sudo mkdir /var/www/apidoc ; sudo mkdir /var/www/visbo-web-ui ; sudo mkdir /var/log/visbo
    ii. sudo chown ubuntu /var/www/apidoc /var/www/visbo-web-ui /var/log/visbo
    iii. sudo chgrp ubuntu /var/www/apidoc /var/www/visbo-web-ui /var/log/visbo
f. Update .bashrc add bashrc from setup folder to the end

21.
a.
    i. sudo npm install pm2@latest -g
    ii. pm2 install pm2-logrotate // no sudo, it runs as user not as root
b.
    i. pm2 list
    ii. pm2 start MyApp/myapp.js
    iii. pm2 set pm2-logrotate:retain 365 // set the Log file expiration to 365
    iv. pm2 save // save the current process list to start on reboot
    v. pm2 stop id/app
    vi. pm2 del id/app
    vii. pm2 startup systemd // configure to start pm2 on reboot
    viii. pm2 unstartup systemd // to remove the startup procedure again

22.
a. echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu bionic/mongodb-org/4.0 multiverse"| sudo tee /etc/apt/sources.list.d/mongodb-org-4.0.list
b. sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 9DA31620334BD75D9DCB49F368818C72E52529D4
c. sudo apt-get update
d. sudo apt-get install -y mongodb-org-shell
e. enable mongo access by setting up peering between the VPC and mongo

23.
a. setup REDIS Server in AWS Elasticache / Redis
    i. assign it to the VPC
    ii. create SubnetGroup
    iii. choose t2.micro REDIS Server

iv. 0 Replications at the moment

v. No Save to Disk

24.

a.

i. adopt environment for ReST Server Configuration dist/env.js

ii. sudo rm -rf /var/www/visbo-web-ui/*; sudo cp -r dist/* /var/www/visbo-web-ui // copy the files to an nginx location after the compilation was successful

iii. generate or configure DKIM Keys for signed e-Mails
please refer to the document DKIM Setup for Send Mail from Visbo ReST Server

b.

i.

1. DB Connection String for ReST Server

ii. pm2 start $HOME/GitHub/visbo-server-rest/startReST.sh --name VisboReST // to start the ReST Server

iii. curl localhost:3484 // test that it is working should show "you should not visit this page"

iv. pm2 startup // to start PM2 after reboot

v. pm2 save // to save the current process list and restart it after reboot

25.

#### Permission Management inside AWS

AWS has a permission management to define who can do what. There are different user management systems inside AWS that has to be distinguished:

1. AWS Organisation
   The main user (who holds the contract and gets the invoices for all what is used in this organisation) can invite additional people and give them access to certain topics (EC2, Invoice & Cost Explorer, ...) and so on. If one of these user creates new resources he is the only one who can see them. This user can then manage permissions and additional users inside IAM (Identity and Access Management) and assign them specific permissions to use the resources.
   The Main Account can also manage an Organisation, I have the impression that this is not required for Visbo. We have started this but somehow it seems to be difficult to get rid of it again.
2. Manage Users, Groups and Policies inside IAM
   Inside the IAM, you can manage a huge amount of permissions and combine it to a policy. This policy can than be assigned to a group. The user could be member of several groups and

gets the combined permission of all policies assigned to the groups where he is member of. AWS supports also negative permissions means a permission is denied. This is used in our case to remove the delete permission for most of the users, to avoid unintended destruction of the system. The deny permission overrules other policies which allow the action for the user.
For Visbo we have setup a VisboReadOnly, VisboManage and a VisboFullAccess Policy that is assigned to the specific groups.

#### Open Topics TODO

1. Setup Patch Management (automated, nightly)