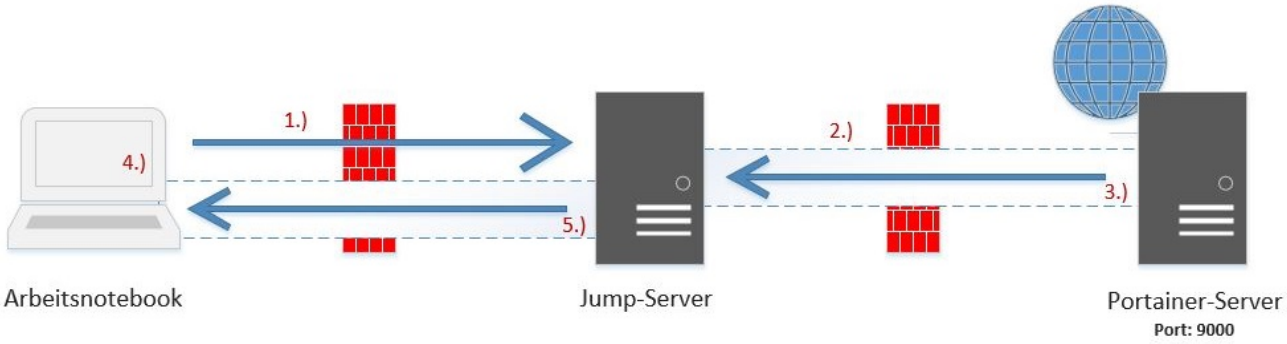
Tunnel Application over SSH

Wenn es Firewall-technisch nicht möglich ist, direkt auf eine bestimmte eingerichtete Applikation auf einem Applikationsservers zuzugreifen; kann versucht werden die Firewalls via SSH-Tunnel zu umgehen.

Als Beispiel habe ich hier den Portainer-Server, welcher in einem separaten Servernetzwerk steht und nur via SSH über den Jump-Server erreicht werden kann ausgewählt. Der Portainer-Server, respektive das Web-GUI läuft unter dem Port 9000.

Gebraucht werden somit zwei Tunnel, in denen wir jeweils den Traffic des Portainer-Servers verschlüsselt durch den Jump-Host hindurch auf den localhost des Arbeitsnotebooks forwarden werden.

Vorgehen zum erstellen der benötigten Tunnel



1.) Aufbau einer neuen SSH Session auf den Jump-Hosts

Als erstes brauchen wir eine neue Session, welche von unserem Arbeitsnotebook auf den Jump-Host aufbauen. **Wichtig: Diese Session (Putty-Fenster) muss später immer geöffnet bleiben, solange wir den Tunnel verwenden möchten!**

```
# ssh vnix1a
```

2.) Erstellen des ersten Tunnels vom Jump-Host auf den Portainer Server

Im zweiten Schritt, wird nun in der soeben erstellten Session einen neuen Tunnel von Jump-Host (vnix1a) auf den Portainer-Server (vredsen2) erstellt.

```
# ssh -L 19000:localhost:9000 vredsen2
```

Details zum Befehl: Es wird via **ssh** eine neue Session mit einem localport **-L 19000** auf dem Jump-Host erstellt. Der Input für diese Schnittstelle, bekommt der Server via Tunnel vom Portainer-Server (vredsen2) über den **localhost:9000**

3.) Zwischen-test des ersten Tunnels

Um zu überprüfen, ob nun der erste Tunnel und die zeitgleich eingerichtete Port-Weiterleitung korrekt funktioniert, kann mithilfe eines CURLs bestimmt werden.

Dazu eröffnen wir eine zweite Session auf den Jump-Host und machen ein CURL auf den lokal weitergeleiteten Port und schauen so, ob die Backend-Applikation auf dem Jump-Host ansprechbar ist.

```
# curl 127.0.0.1:19000 | grep portainer
```

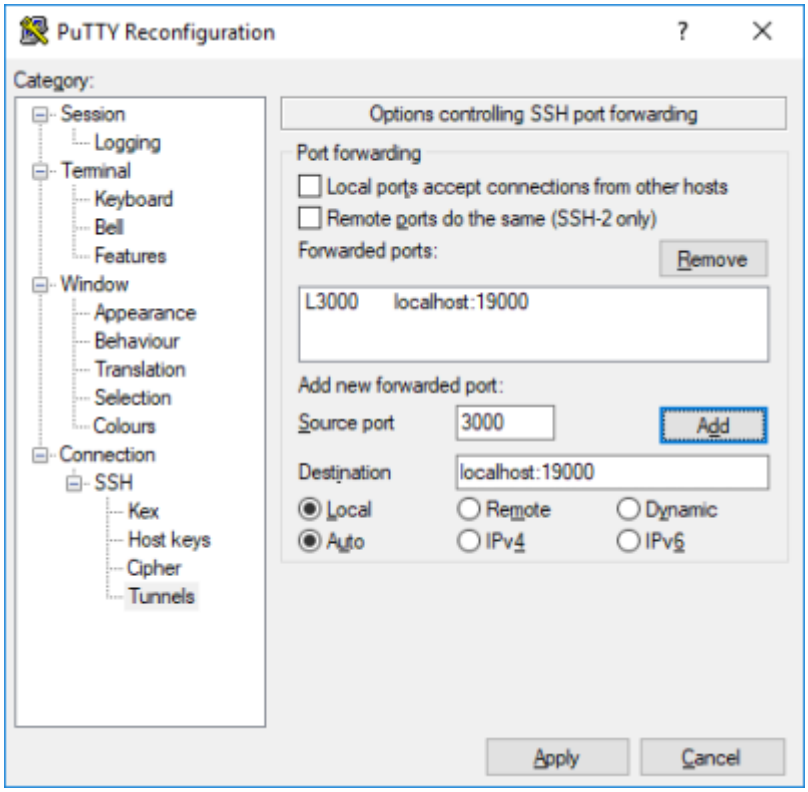
```
[rebermi@vnix1a ~]$ curl 127.0.0.1:19000 | grep portainer
  % Total    % Received % Xferd  Average Speed   Time    Time     Time
Current
                                 Dload  Upload   Total   Spent    Left
Speed
100  2748  100  2748    0     0   486k      0 --:--:-- --:--:-- --:--:--
536k
<html lang="en" ng-app="portainer">
    open: toggle && ['portainer.auth', 'portainer.init.admin',
'portainer.init.endpoint'].indexOf($state.current.name) === -1,
    nopadding: ['portainer.auth', 'portainer.init.admin',
'portainer.init.endpoint'].indexOf($state.current.name) > -1 ||
applicationState.loading
```

So wie es aussieht, funktioniert der erste Tunnel! ;)

4.) Erstellen des zweiten Tunnels vom Arbeitsgerät auf den Jump-Host

Damit unserer Tunnel schlussendlich auch durchgehend bis auf unser Arbeitsnotebook funktionieren kann, wird noch der zweite Tunnel benötigt.

Um diesen zu erstellen, muss ein weiteres Putty geöffnet werden, mit dem wir anschliessend auf den Jump-Host verbinden. Jedoch werden hier noch vor dem eigentlichen verbinden folgende Anpassungen vorgenommen.



Nach dem Anpassen kann die Session eröffnet werden und der zweite Tunnel sollte nun stehen!

5.) Finaler Test beider aufgebauten Tunnel

Im Punkt 4. haben wir auf unserem Arbeitsgerät einen neuen Listener auf Port 3000 erstellt. Die Informationen welche an diesen Listener geschickt werden, bekommen wir vom Jump-Host via Port 19000 mitgeteilt. Es solte nun also ein durchgehender Tunnel stehen, durch welchen wir auf unsere Applikation im Servernetz zugreifen könnten.

Testen: → Im Browser: http://127.0.0.1:3000

Last update: **2018/08/27 08:06**