Mitarbeiten an invis-Server

Sie haben Interesse sich im Projekt zu engagieren? Wenn ja hier eine kurze Anleitung zum Einstieg.

Neben der reinen Entwicklungsarbeit steht natürlich auch die Arbeit an der Doku hier im Wiki sowie die Vorstellung des Projektes auf verschiedenen Open-Source-Events an. Letzteres kann eher als Belohnung, denn als Arbeit verstanden werden. 

Voraussetzungen

Wer mitarbeiten möchte sollte grundlegende Kenntnisse in Sachen Linux und Netzwerke mitbringen, keine Angst vor Shellscripts haben und/oder (darüber würden wir uns am meisten freuen) Kenntnisse in PHP und JavaScript haben. Kenntnisse im RPM-Paketbau sind ebenfalls sehr willkommen.

Für die Mitarbeit benötigen Sie Zugänge zu **Github**, dem **openSUSE Build Service** und **openSUSE Projektverwaltung "Progress"**. Die Registrierung ist jeweils kostenlos. Der Build Service und Progress verwenden das gleiche Authentifizierungsbackend. Eine Registrierung genügt also um Zugang zu beiden Systemen zu erhalten.

Wenn Sie sich dort registriert haben, melden Sie sich bei uns. Wir sollten das eine oder andere Gespräch führen um uns gegenseitig zu beschnuppern. Sie können Sich auch gerne bereits im Vorfeld die Quellen des Projektes von Github herunter laden und uns mit der einen oder anderen Verbesserung überraschen.

Kontakt: info(at)invis-server.org

Bevor Sie aktiv mitarbeiten sollten Sie bereits den einen oder anderen invis-Server installiert und sich damit vertraut gemacht haben.

Wenn wir miteinander klar kommen, tragen wir Sie auf den beiden genannten Plattformen als Projektmitglied ein. Ab diesem Zeitpunkt können Sie aktiv und selbständig am Projekt mitarbeiten.

Zusätzlich sind derzeit noch zwei Mailinglisten für Entwickler und Nutzer aktiv. Zur deren können Sie sich unter folgender URL eintragen: https://ml.invis-server.org/mailman/listinfo/invis-dev. Der Eintrag in die Entwickler-Liste erfordert allerdings unsere Bestätigung.

Mittelfristig wollen wir die Mailinglisten aber zu Gunsten der Forenfunktion in Progress aufgeben, registrieren Sie sich also bitte nicht mehr in den Mailinglisten.

Github & OBS

Grundsätzlich gehen die Arbeiten an den Quellen bei Github und der zugehörigen Aktualisierung des invis-setup RPM-Paketes Hand in Hand. D.h. Wird etwas an den Quellen geändert wird die Änderung ins RPM übernommen.

Das bedeutet, dass in einer Testumgebung sowohl die Quellen des Github-Repositories als auch die aktuellste Version des RPMs vorhanden sein sollten.

Github

Installieren und konfigurieren Sie also am einfachsten eine virtuellen Maschine, gemäß der Installationsanleitung hier aus dem Wiki. Sie sollten das invis-Setup RPM installieren, allerdings ***sine*** noch nicht starten. Installieren Sie zusätzlich ***git*** auf dieser Maschine.

```
linux:~ # zypper in git
```

git einrichten und benutzen

Nach der Installation von ***git*** sollten Sie ein paar grundlegende Dinge einrichten:

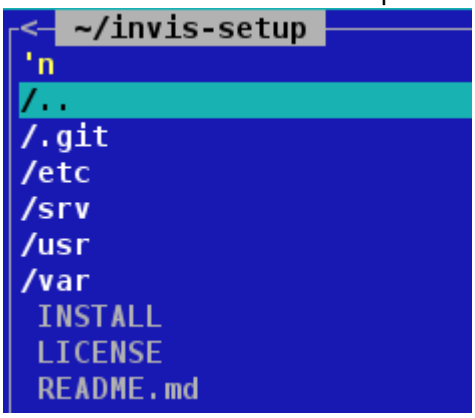
```
linux:~ # git config --global push.default simple
linux:~ # git config --global user.name "Vorname Nachname"
linux:~ # git config --global user.email "email Adresse"
```

invis Quellen herunterladen

Danach können Sie das invisAD-Repository in Ihre Entwicklungsumgebung klonen:

```
linux:~ # git clone https://github.com/invisserver/invisAD-setup.git
```

Im geklonten Repository befinden sich neben dem Git-Arbeitsverzeichnis und ein paar Readme-Dateien die Verzeichnisse /etc, /usr, var und /srv. Sie bilden die Basis für den Bau des invis-setup RPM-Paketes.



Die Inhalte der Verzeichnisse entsprechen genau der Lage der Dateien und Verzeichnisse nach der Installation des RPM-Paketes und dürfen nicht (bzw. nur nach Absprache) geändert werden.

So sind die Shellscripts der invis-Toolbox wie auch das Setup-Script ***sine*** unter

```
/usr/bin
```

zu finden und die Vorlagen der Konfigurationsdateien des Server-Setups unter:

```
/usr/share/doc/packages/invis-setup/examples
```

Änderungen übertragen

Wurden Änderungen im Repository vorgenommen, müssen diese übernommen und später mit dem zentralen Repository bei Github abgeglichen werden:

Zunächst können die vorgenommenen Änderungen wie folgt aufgelistet werden:

```
linux: ~/invis-setup # git status
```

Entsprechend der Ausgabe des vorangegangenen Befehls müssen neue bzw. geänderte Dateien ins Repository aufgenommen werden:

```
linux: ~/invis-setup # git add /pfad/zur/datei
```

Gelöschte Dateien müssen entfernt werden:

```
linux: ~/invis-setup # git rm /pfad/zur/datei
```

Danach folgt der sogenannte Commit, bei dem Änderungen mit einem Kommentar versehen werden:

```
linux: ~/invis-setup # git commit
```

Der Befehl öffnet den (allseits beliebten) Editor ***vi*** und erwartet die Eingabe eines Kommentars.

Beim Kommentieren sollte in kurzen Worten die vorgenommene Veränderung beschrieben werden. Bitte „committen“ Sie Ihre Veränderungen in kleinen Schritten. Einen ganzen Tag an allem basteln und dann einen einzelnen Commit machen mit einem Kommentar wie *„jetzt funktioniert alles“* ist nicht gerade für eine produktive Zusammenarbeit geeignet.

Danach kann die Veränderung mit dem zentralen Repository abgeglichen werden:

```
linux: ~/invis-setup # git push
```

Beim Hochladen werden Sie nach Ihren Github-Zugangsdaten gefragt.

Wenn Sie nach ein paar Tagen Pause mit Ihrem Repository-Klon weiterarbeiten möchten, ist es **sehr wichtig**, dass sie zunächst in die andere Richtung abgleichen:

```
linux: ~/invis-setup # git pull
```

Es könnte ja sein, dass inzwischen jemand anderes weitergearbeitet hat.

Arbeiten mit Branches

Mit dem Umbau unserer Build-Service Strukturen haben wir begonnen in Github Branches für stabile Versionen des invis-Server Setup-Paketes anzulegen. Das bedeutet, dass die grundlegende Entwicklungsarbeit immer am „Master Branch“ geleistet wird. Um aber auch an einem als stabil gekennzeichneten Paket Bugfixes vorzunehmen muss vom Master auf den jeweiligen Branch gewechselt werden.

Der Wechsel ist denkbar einfach. Hier am Beispiel des Branches „Stable_10.1“. Grundsätzlich enthält jedes Repository nach dem es geklont wurde alle Branches. D.h. es kann lokal immer zwischen den Branches gewechselt und an ihnen gearbeitet werden.

Nach dem Klonen befindet man sich immer im Master-Branch, ein Wechsel erfolgt mit:

```
linux:~/invisAD-setup # git checkout Stable_10.1
```

Jetzt können an diesem Branch Veränderungen vorgenommen und per Commit aufgenommen werden. Es kann aus dieser Situation in gleicher Weise auf jeden anderen Branch gewechselt und daran gearbeitet werden. Ein abschliessendes:

```
linux:~/invisAD-setup # git push
```

überträgt alle vorgenommenen Änderungen zurück an Github.

OBS

Bevor Sie sich als „Frischling“ an den Buildservice wagen, sollten Sie sich ein wenig mit dessen Spielregeln und dem RPM-Paketbau an sich auseinandersetzen. Der Umgang damit ist alles andere als trivial.

Einen guten Einstieg ins Thema finden Sie hier:

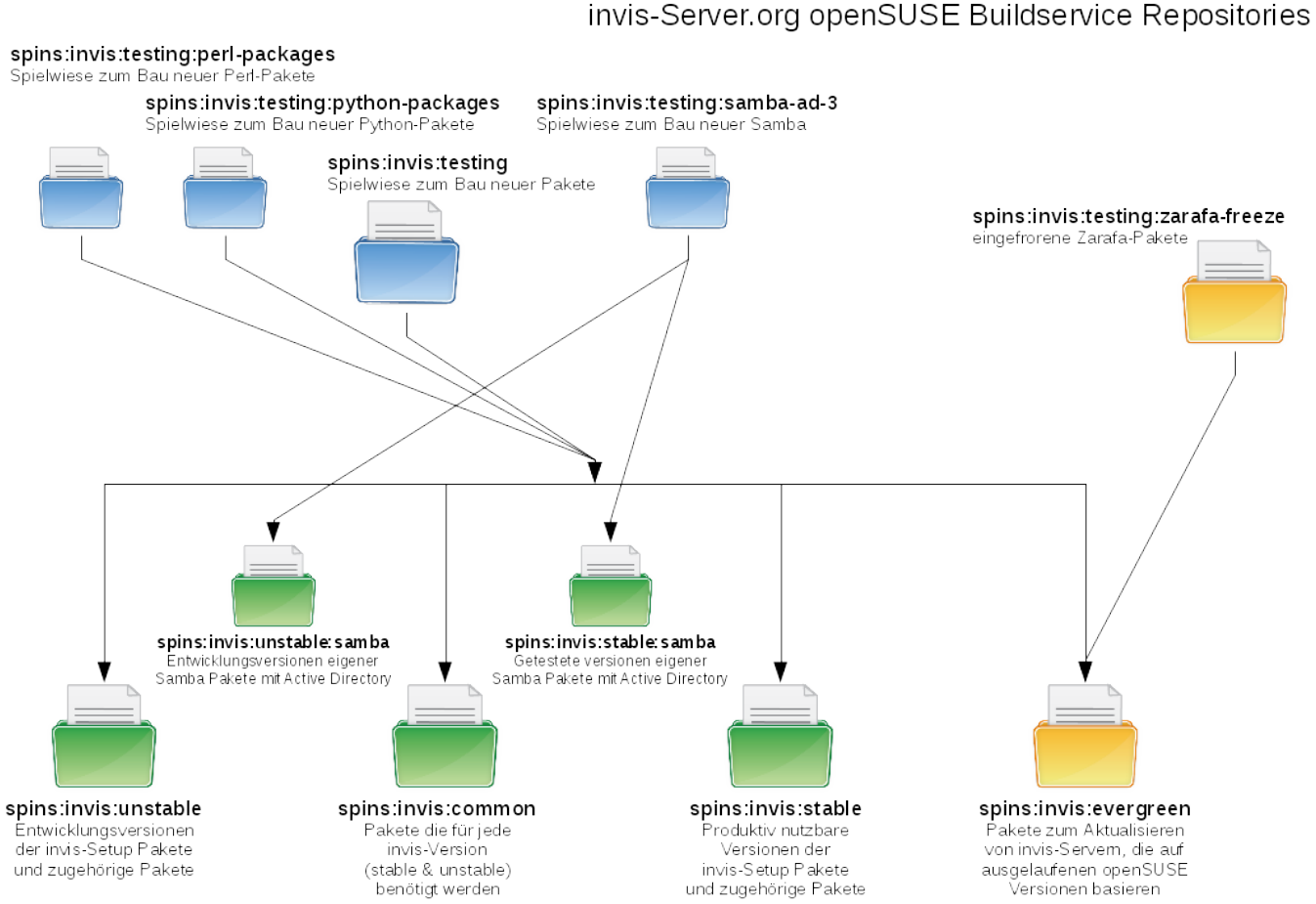
- **http://de.opensuse.org/Portal:Build_Service**
- **http://de.opensuse.org/Portal:Paket-Management**
- **http://de.opensuse.org/Portal:Paketbau**

Lassen Sie sich nicht durch die Fülle an Informationen entmutigen. In dem Moment, wo Sie sich am OBS registriert haben, verfügen Sie über ein so genanntes „Home-Project“, dort können Sie erste Gehversuche in Sachen Paketbau unternehmen.

invis Server Repositories im OBS

invis-Server wird vom openSUSE-Projekt als sogenanntes „Spin-Off“ geführt. Entsprechend verfügt das Projekt mit **„spins:invis“** über einen offiziellen Repository-Zweig im openSUSE Buildservice.

Unterhalb dieses Haupt-Repositories haben wir eine Reihe von Subrepositories angelegt, die zum Einen der Paket-Entwicklung als auch als Basis für die Installation von invis-Servern dienen. Hier der Überblick (Stand Januar 2017):



Die in der Abbildung **blau** dargestellten Repositories dienen der Paket-Entwicklung. Die Ausgliederung von Perl- und Python-Paketen in eigene Subrepositories dient dabei schlicht der Übersicht. Bei beiden Script-Sprachen wächst die Anzahl der Pakete durch Paketabhängigkeiten untereinander sehr schnell an. Mit eigenen Subrepositories ist es einfacher die Übersicht zu behalten.

Die **grün** dargestellten Repositories dienen als Basis für die Installation von invis-Servern, sie werden aus den Testing-Repositories gespeist. In den grün dargestellten Repositories werden keine Veränderungen an den Paketen vorgenommen. Die Subrepositories **stable** und **unstable** unterscheiden sich dahingehend, dass Pakete in **stable** nicht automatisch erneuert werden, wenn an deren Urpsrungs-Paketen in den Testing-Repositories Änderungen vorgenommen werden. In **unstable** dagegen hat eine Änderung des Ursprungspaketes im Testing-Zweig sofort einen Neubau des korrespondierenden Paketes zur Folge.

Die **gelb** dargestellten Reposietories dienen der Langzeitpflege von invis-Servern deren zugrunde liegende openSUSE-Version bereits „out of maintenance“ ist. Der Paketumfang hier entspricht nicht zwingend dem der grün dargestellten Repositories. Eine Neuinstallation aus diesen Repositories ist nicht möglich und auch nicht angedacht.

Für diejenigen im invis-Team, die an der Paketpflege beteiligt sind gelten beim Paketbau ein paar Spielregeln:

1. Neue Pakete werden grundsätzlich in **spins:invis:testing** oder je nach Zugehörigkeit in dessen Subrepositories „:perl-packages“ und „:python-packages“ gebaut. Dabei spielt es keine Rolle, ob die Pakete vollständig von uns gepflegt oder sie aus einem fremden Repository im Buildservice „gebranched“ sind.
2. Pakete die in den Installationsrepositories benötigt werden, werden immer als Branch auf das Ursprungspaket in **spins:invis:testing** oder dessen Perl- und Python Subrepositories angelegt.

Ein direktes erzeugen neuer Pakete hier findet nicht statt, ausgenommen derjenigen deren Code wir selbst in Github pflegen. Dies sind derzeit „invis-setup“ und „invisAD-setup“.

3. Pakete deren Code wir selbst in Github pflegen werden zunächst in **spins:invis:testing** gebaut. Sie werden hier solange entwickelt, bis erfolgreich funktionierende RPM Pakete vorliegen. Ist dies erledigt ziehen die Pakete zunächst nach **spins:invis:unstable** um und können von dort aus installiert und getestet werden. Nach erfolgreichen Tests legen wir den Zeitpunkt fest, ab dem eine Version als Stable-Release gilt und erzeugen einen entsprechenden Branch in Github. Auf Basis dieses Branches wird dann ein Paket in **spins:invis:stable** erzeugt.
4. In die Repositories **spins:invis:stable** und **spins:invis:unstable** werden nur Pakete eingebracht, die vollständig von uns selbst maintained, oder zwingend für die Installation des invis-Setup-Paketes benötigt werden. Alle anderen Pakete gehören nach **spins:invis:common**
5. Wird ein Paket in **spins:invis:stable** gebranched muss **immer** die Option ***Stay on current revision, don't merge future upstream changes automatically*** gesetzt werden. Dies verhindert, dass durch Weiterentwicklung beschädigte Pakete im Stable-Repository landen. (Wer einen solchen Branch mit dem Kommandozeilenwerkzeug ***osc*** anlegt muss die Option ***-c*** verwenden (Beispiel: osc linkpac -c spins:invis:testing group-e spins:invis:stable)
6. Für den Bau von Samba-Paketen mit Active-Directory wurde eine eigene Repository-Struktur geschaffen. Sie besteht aus **spins:invis:testing:samba-ad-3**, **spins:invis:stable:samba** und **spins:invis:unstable:samba**. Es gelten die gleichen Spielregeln wie oben beschrieben. D.h.: Pakete in **spins:invis:stable:samba** aktualisieren sich nicht automatisch, wenn sich im Testing Repo etwas ändert.
7. Es werden **keine** Pakete direkt in **spins:invis** gepflegt.

Kurzanleitung zum Aktualisieren eines invis-Setup Paketes

Haben Sie am invis-server Quellcode nur kleine Veränderungen vorgenommen, ist die Vorgehensweise zur Übernahme ins RPM-Paket einfach und überschaubar.

Legen Sie auf Ihrer Testmaschine ein Verzeichnis unter dem Namen „invis-setup-*Versionsnummer*“ bzw. „invisAD-setup-*Versionsnummer*“ an. Derzeit lauten die Versionsnummer **9.2** beim Classic-Server und **10.0** bei der ActiveDirectory Variante.

```
linux:~ # mkdir invis-setup-9.2
```

Kopieren Sie aus dem Klon des Github-Repositories die Verzeichnisse /etc, /usr, /srv und /var in dieses Verzeichnis.

```
linux:~/invis-setup # cp -r etc/ usr/ srv/ var/ ~/invis-setup-9.2/
```

Packen Sie das neue Verzeichnis in einem tar.gz Archiv zusammen und laden Sie es dann in das Testing-Repository des invis-Server Projektes hoch.

```
linux:~ # tar -czf invis-setup-9.2.tar.gz invis-setup-9.2/
```

Hinweis: *Ab Version 11.0 wurde die Major-Release-Nummer mit in den Paketnamen aufgenommen. Dies erfordert eine Umbenennung des Quellcode-Archivs:*

```
linux:~ # mkdir invisAD-setup-11-11.0
```

```
linux:~/invis-setup # cp -r etc/ usr/ srv/ var/ ~/invisAD-setup-11-11.0/
```

```
linux:~ # tar -czf invis-setup-9.2.tar.gz invisAD-setup-11-11.0/
```

Wenn Sie sich unsicher sind, ob sie korrekt vorgehen, bitten Sie andere Projektmitglieder um Hilfe. Es sollten jedenfalls nur erfolgreich getestete Änderungen Ihren Weg ins RPM-Paket finden.

Kleine strukturelle Änderungen sollten keine Änderungen am Specfile des RPM-Paketes erforderlich machen.

Ist das neue Quell-Archiv gepackt kann es in den Buildservice, genauer ins Repository **„spins:invis:testing“** hochgeladen werden. Änderungen dort ziehen umittelbar auch ein erneutes Bauen des Paketes im Repository **„spins:invis:unstable“** nach sich. Aus diesem Repository heraus kann dann eine oder mehrere Testinstallation(en) folgen. Funktioniert alles wie gewünscht ist ein neuer Branch des Paketes in **„spins:invis:stable“** erforderlich. Dabei ist darauf zu achten, dass dabei **immer** die Option:

Stay on current revision, don't merge future upstream changes automatically

aktiviert wird. Dies verhindert, dass nocht zu testende neue Pakete automatisch in **„spins:invis:stable“** gebaut werden.

Aktualisieren des invis-Setup Paketes per _service-file

Neben der oben beschriebenen Variante der Aktualisierung testen wir aktuell (19.2.2017) die automatische/halbautomatische Aktualisierung per OBS-Service. Ein OBS-Service kann z.B. ein bestimmtes Github-Repositorie klonen und mit bestimmen Verzeichnissen aus dem Klon ein tar erstellen:

```
<service name="tar_scm">
  <param name="scm">git</param>
  <param name="url">git://github.com/invisserver/invisAD-setup.git</param>
  <param name="subdir"></param>
  <param name="include">etc</param>
  <param name="include">srv</param>
  <param name="include">usr</param>
  <param name="include">var</param>
  <param name="filename">invisAD-setup-12</param>
  <param name="versionprefix"></param>
  <!-- Holt unstable -->
  <param name="version">12.1</param>
  <param name="revision">master</param>
  <!-- Holt Stable -->
  <!--
  <param name="version">12.0</param>
  <param name="revision">Stable_12.0</param>
  -->
</service>
```

Im weiteren Verlauf kann dieses tar gezippt werden:

```
<service name="recompress">
  <param name="file">*invisAD-setup*.tar</param>
  <param name="compression">gz</param>
</service>
```

Zum Schluß wird die Versionsnummer im *.spec noch automatisch angepasst:

```
<!-- Setzt Version in spec-file -->
<service name="set_version"/>
```

Getriggert wird dieser Service entweder über das Webinterface mit dem „Trigger Services“ Button oder per osc mit dem Befehl:

```
osc service remoterun spins:invis:testing invisAD-setup-12
```

Als Rückgabe sollte ein „Ok“ kommen. Den aktuellen Zustand sieht man entweder im Webinterface oder auch wieder per osc:

```
osc results spins:invis:testing invisAD-setup-12
```

bzw.

```
osc results spins:invis:unstable invisAD-setup-12
```

Die Rückgabe „succeeded“ (ohne Sternchen) bedeutet der Build ist fehlerfei durchgelaufen und man kann das neue Paket installieren.

progress.openSUSE.org

Hinter der openSUSE Projektverwaltung steckt die recht verbreitete Software Redmine. Software Entwicklern wird sie vermutlich bekannt vorkommen. Sie verbindet verschiedenste Funktionen wie etwa ein Ticket-System, Projektverwaltung, Diskussionsforen uvm.

Innerhalb des Systems bildet „invis-Server“ unser Hauptprojekt. In diesem Projekt sind die beiden Diskussionsforen angesiedelt:

1. **Entwickler-Forum:** https://progress.opensuse.org/projects/invis-server/boards/12
2. **Anwender-Support-Forum:** https://progress.opensuse.org/projects/invis-server/boards/15

Einrichten einer Testumgebung

Als Entwicklungs- und Testumgebung eignet sich eine Umgebung zweier virtueller Maschinen am besten. Als Virtualisierungumgebung empfehlen wir aufgrund der einfachen Handhabung und der Verfügbarkeit für verschiedene Betriebssysteme **Virtualbox** .

Vorbereitung VMs

Installieren Sie Virtualbox am besten inklusive VirtualBox Extension Pack und richten Sie darin zwei virtuelle Maschinen ein. Davon eine Windows-Maschine, idealerweise ab Windows 7 aufwärts und eine openSUSE Maschine derzeit in Version 13.1 (bitte noch nicht 13.2 verwenden).

Bei der Einrichtung der VMs müssen der Linux Maschine zwei Netzweradapter konfiguriert werden. Davon sollte der erste Adapter entweder als NAT Device, oder besser als Netzwerkbrücke und die zweite Netzwerkkarte als „internes Netzwerk“ eingerichtet werden. Für das interne Netzwerk können Sie nach eigenem Geschmack einen Namen vergeben. Dieses „benannte Netzwerk“ funktioniert wie ein virtueller Switch an den weitere VMs angeschlossen werden können.

Führen Sie dann auf der openSUSE VM eine ganz normale invis-Server Installation durch, wie hier im Wiki beschrieben.

Die Windows-Maschine bekommt nur einen Netzwerkadapter, der an das zuvor generierte „interne Netzwerk“ angebunden werden muss. Internetzugang erhält diese Maschine über den als Router arbeitenden virtuellen invis-Server. Wichtig ist, dass Sie auf der Windows VM (auch wenn es lange dauert) alle anstehenden Updates einspielen.

Installieren Sie auf der Windows VM die Microsoft **Remote Server Tools** . Sie benötigen die Tools zur Administration des Active Directories.

Einrichten der MS Remote Server Tools

Bei den Remote Server Tools handelt es sich nicht um eine eigenständige Software, sondern um ergänzende Windows-Funktionen, die Sie über die Systemsteuerung aktivieren müssen.

Freundlicherweise blendet der Installer gleich nach der Installation die Windows-Hilfe ein, in der die Aktivierung der neuen Funktionen erläutert werden. Hier in aller Kürze:

1. Sytemsteuerung → Programme und Funktionen öffnen.
2. Dann im linken Rand „Windows Funktionen aktivieren oder deaktivieren“ anklicken
3. Aktivieren Sie unter „Remoteserver-Verwaltungstools“ → „Featureverwaltungstools“ den Punkt „Tools für die Gruppenrichtlinienverwaltung“
4. … und unter „AD DS- / AD LDS-Tools“ → „AD DS-Tools“ die Punkte „Active Directory-Verwaltungscenter“ und „Server für NIS Tools“
5. … und dann noch unter „AD DS- / AD LDS-Tools“ den Punkt „DNS Server-Tools“

Mit weiteren Funktionen können Sie ja nach belieben experimentieren, die genannte Auswahl genügt jedenfalls zur Administration des ADs inklusive Nameserver.

Zur Nutzung der Tools führen Sie mit der Windows-VM einen Domänenbeitritt durch und melden sich dann mit dem Domänen-Administrator an. Sie finden die Tools im Startmenü. Nach dem Start der jeweiligen Tools müssen Die jeweils den vollständigen Hostnamen des invis-Servers ageben.

invis VM

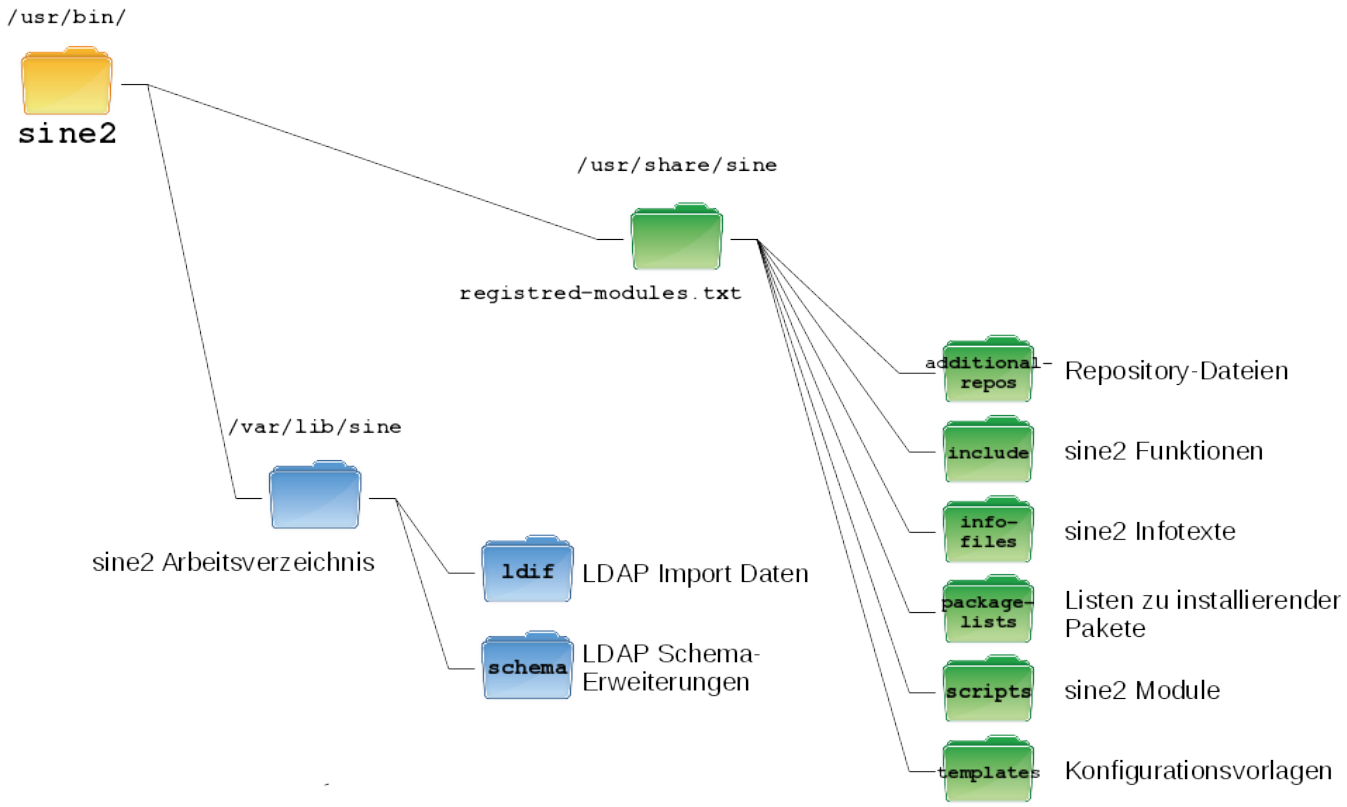
Auf der invis-Server VM sollten Sie wie oben Beschrieben zusätzlich zum installierten RPM noch das

GIT-Repository klonen. Danach steht dem Testen und Entwickeln nichts mehr im Weg.

Aufbau "sine2"

Im Vergleich zu seinem Vorgänger hat sich die Struktur von ***sine2*** deutlich geändert. Aufgrund der viel zu großen Zeilenanzahl der ersten sine-Version (ca. 3500 Zeilen) wurde mit ***sine2*** auf eine modularisierte Version des Setup-Scripts umgestellt. Die daraus resultierenden Einzel-Scripts der verschiedenen sine-Module sind für sich genommen jetzt überschaubar kurz geworden. Dies wurde allerdings erkauft durch einen komplexeren Aufbau der Gesamtstruktur.

Um an ***sine2*** mitarbeiten zu können, muss die neue Struktur natürlich verstanden werden, daher hier ein Überblick:



Zu ***sine2*** gehören zwei Verzeichnisstrukturen, das Arbeitsverzeichnis unter **`/var/lib/sine`** und das Ressourcen-Verzeichnis unter **`/usr/share/sine`**.

Arbeitsbereich

Hier landen Dateien, die ***sine2*** während des Setups eines invis-Servers dynamisch generiert. Diese Dateien sollten Sie, außer Sie wissen, was Sie tun, auf keinen manuell ändern oder löschen. Sie beeinflussen damit unmittelbar das Verhalten des Setups.

- **`/var/lib/sine/ldif`** - In diesem Verzeichnis werden LDIF Dateien generiert und an die Installationsumgebung angepasst um Sie anschließend in das LDAP-Verzeichnis des Active-Directory zu importieren. Darunter beispielsweise sämtliche Links des invis-Portals.
- **`/var/lib/sine/schema`** - Um eigene Informationen, wie beispielsweise die Daten des DHCP-

Servers im AD speichern zu können müssen LDAP Schema-Erweiterungen eingespielt werden. Diese werden hier an Ihre AD-Struktur angepasst.
- **`invis_confdata`** - Diese Datei enthält alle von ***sine2*** abgefragten Konfigurationsdaten eines invis-Servers. Darunter auch Passwörter. Diese Datei sollte nach der Installtion gesichert und dann gelöscht werden.
- **`prepstat`** - Diese Datei enthält lediglich Namen und Nummer des als nächstes auszuführenden ***sine2*** Moduls.
- **`sine_temp`** - Ist eine Datei, die die Inhalte einer Abfrage des in sine2 verwendeten Tools „dialog“ enthält. Das räumt ***sine2*** zum Schluss selbst auf.
- **`etc-backup-*-tar.gz`** - Ist eine Sicherung des **`/etc`** Verzeichnisses, die ***sine2*** zu Beginn des sysprep-Moduls anlegt.
- Ansonsten liegen dort ggf. noch weitere Dateien, die von ***sine2*** bearbeitet werden.

Ressourcen-Bereich

Hier liegen die einzelnen Komponenten von ***sine2***, hier ist auch der Ort an dem bei der Weiterentwicklung von ***sine2*** der größte Teil der Arbeit anfällt.

- **`registered-modules.txt`** - In dieser Datei werden alle ***sine2*** Module registriert. Die einzelnen Module werden hier, in korrekter Reihenfolge in der sie abgearbeitet werden, eingetragen. Die Datei ist 3-spaltig. Die erste Spalte enthält eine 3-Stellige Nummer des Moduls, es folgt in Spalte 2 die Kennzeichnung, ob es sich um ein „Pflichtmodul“ (d) oder ein optionales Modul (o) handelt. Die letzte Spalte enthält den Namen des Moduls. Das Trennzeichen ist der Doppelpunkt.
- **`/usr/share/sine/additional-repos`** - Hier werden Repository-Dateien für zusätzliche Software, die nicht Teil der openSUSE Distribution ist, hinterlegt.
- **`/usr/share/sine/include`** - Hier liegen Dateien, die in ***sine2*** oder in die zugehörigen Modul-Scripte inkludiert werden können. Derzeit beschränkt sich dies auf die Datei **`functions`** die immer wieder benötigte Script-Funktionen enthält.
- **`/usr/share/sine/infofiles`** - Hier liegen Textdateien, die die Informationstexte enthalten, die ***sine2*** während der Installation anzeigt.
- **`/usr/share/sine/package-lists`** - Die hier liegenden Dateien enthalten Listen zu installierender Software-Pakete.
- **`/usr/share/sine/scripts`** - Das Verzeichnis enthält die Einzelscripte der verschiedenen ***sine2*** Module. Diese Scripts sind nur lauffähig, wenn Sie aus ***sine2*** heraus aufgerufen werden.
- **`/usr/share/sine/templates`** - Hierin enthalten ist je ein Unterverzeichnis für jedes Modul. Darin wiederum liegen Vorlagen-Dateien für Konfigurationen einzelner Dienste die von ***sine2*** installiert und eingerichtet werden.

Stand der Dinge / To do

Open-Source-Projekte werden quasi nie fertig. Es ergeben sich ständig neue Anforderungen und Aufgaben für das Projekt, die meist in Form von Teilprojekten und -aufgaben abgearbeitet werden können. Alle anstehenden und in Bearbeitung befindlichen aufgaben werden über openSUSEs Projektverwaltungssystem "Progress" organisiert. Um einen Einblick zu erhalten registrieren Sie sich dort. Die Registrierung ist wie gesagt kostenfrei und verpflichtet **nicht** zur Mitarbeit.