Syncany

Secure open-source file sync

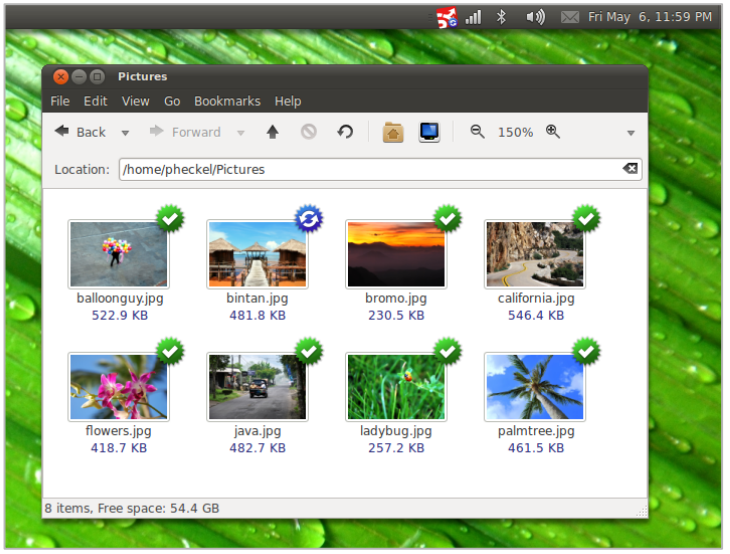
Contact the Syncany Team
via team@syncany.org

Agenda

- **What is Syncany?**
- How does it work?
- Focus Topics:
  - Deduplication & Versioning
  - Cryptography Concept
  - Continuous Integration & Packaging

What is Syncany?

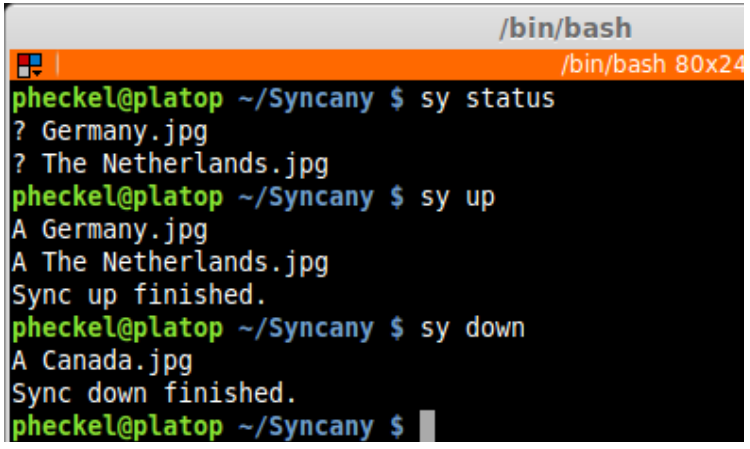
Syncany is an open source Dropbox-like file sync and backup tool. It synchronizes files and folders between computers either manually or automatically. Users can define certain folders on their machine and keep them in sync with friends or colleagues.



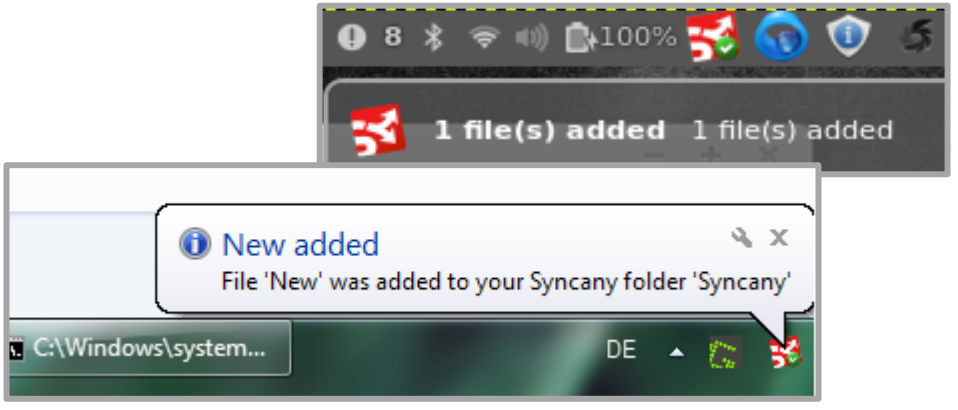
So what makes Syncany **awesome**? What makes it **different**?

- Syncany is **open-source** software
- **Use-your-own** storage (FTP, S3, WebDAV, ...)
- Files are **encrypted** before upload (don't trust anybody but yourself)
- Files are **deduplicated** locally (space savings on remote storage)
- Files are intelligently **versioned** (restoring old versions is easy)

Manual version control-like sync



Automatic Dropbox-like sync



Storage Plugins

> Use-your-own storage, or even better: use *any storage*!

Storage plugins:

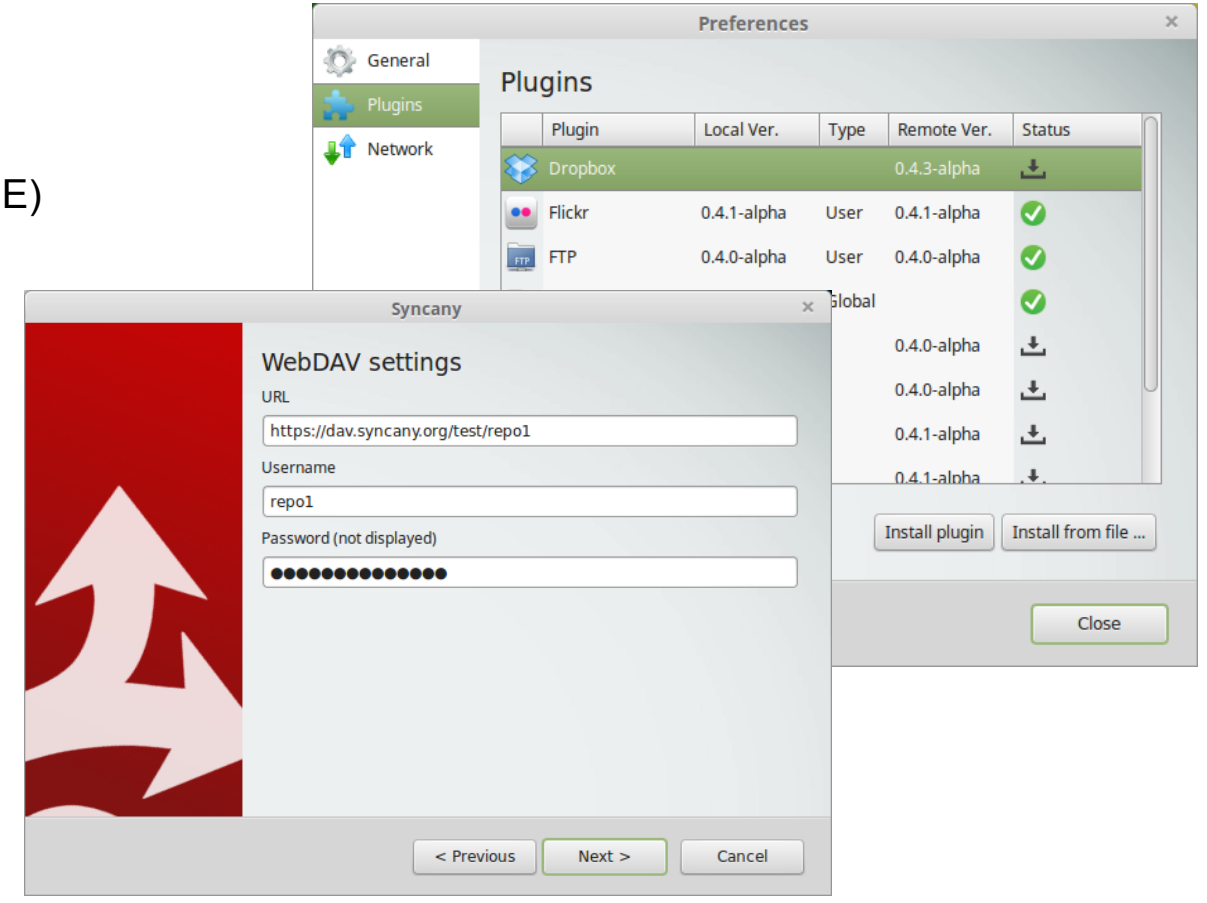
- Amazon S3
- Dropbox
- Flickr
- FTP
- RAID0
- Local (mounted drives; e.g. via FUSE)
- OpenStack Swift
- SFTP
- Samba / CIFS
- PHP / REST (3rd party, incubating)
- WebDAV

(Easy to develop your own
in about 200-300 lines of code)

Other plugins:

Graphical User Interface

Simple Web Interface (PoC, looking for dev)



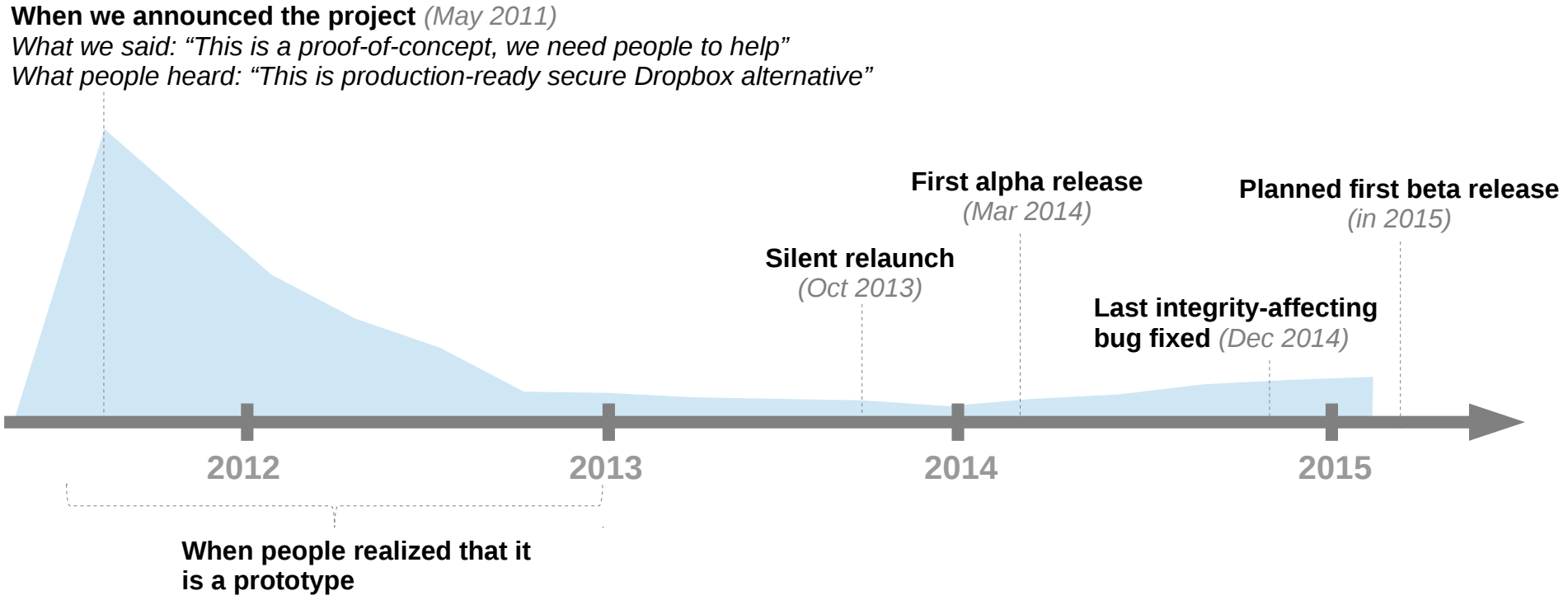
Project History

First attempt: *~ What I learned: Never advertise a prototype!*

- ➔ Started in 2011, announcement of PoC on WebUpd8 (popular blog) and Ubuntu Podcast in May 2011
- ➔ 15,000 visitors in announcement week, 400 developers in mailing list, translated in 30 languages by excited helpers, >2,200 followers on Twitter; then big drop in interest due to non-functioning software.

Second attempt: *~ Doing it right: Make a stable core, then build on top of that!*

- ➔ Silent relaunch in October 2013 (no new announcement yet)
- ➔ 3-4 active developers, ~30 contributors, IRC channel with 25-30 people

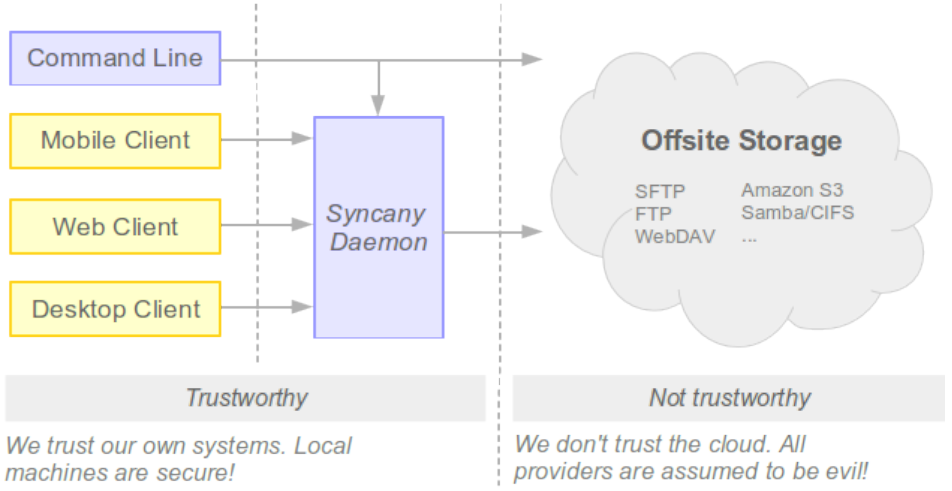


Agenda

- What is Syncany?
- **How does it work?**
- Focus Topics:
  - Deduplication & Versioning
  - Cryptography Concept
  - Continuous Integration & Packaging

How does it work?

High Level Architecture



Components

- **Core library:** Behavior logic (e.g. up/down operation), persistence, plugin architerture
- **Command line interface (CLI):** `$ sy <up|down|..>`
- **Background daemon:** REST/WS API for GUI/Web/Mobile interface
- **Plugins:** Storage plugins & user interface plugins

Core Operations

	CLI	Daemon/GUI
UpOperation *~ part of core library* Analyze → Index → Pack → Encrypt → Upload	`$ sy up`	*When local changes occur & every 2min*
DownOperation *~ part of core library* Analyze → Download → Decrypt → Unpack → Apply	`$ sy down`	*When remote changes occur & every 2 min*

Agenda

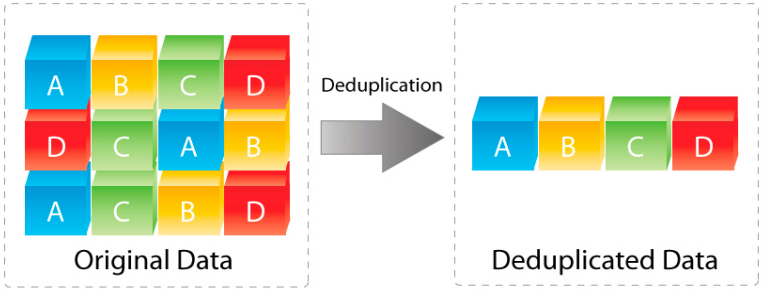
- What is Syncany?
- How does it work?
- Focus Topics:
  - **Deduplication & Versioning**
  - Cryptography Concept
  - Continuous Integration & Packaging

Deduplication & Versioning

> Syncany uses source-side, in-line, fixed-size deduplication

Data deduplication identifies duplicate sequences of bytes (chunks) and only stores **one copy of that sequence**.

If a chunk appears again in the same file (or in another file), deduplication only **stores a reference to this chunk** instead of its actual contents.



Distinction to compression
Not bound to location, date or time of files; stateful approach: local database/index necessary

Characteristics

- Location: Target-side vs. **Source-side**
- Time: **In-line** vs. Post-process
- Chunking method: whole file vs. **fixed-size** vs. variable-size vs. format-aware chunking

Highlighted: Concepts in Syncany

Chunking methods
Methods have different impact on performance, I/O, CPU and RAM usage

- Whole file chunking: Only deduplicate entire files, files are not split into chunks.
- Fixed-size chunking: Files are split into chunks at fixed offsets, e.g. at 32 KB, 64 KB, 96 KB, …
- Variable-size chunking: Files are split into chunks using "*sliding window* "-based chunking method (such as TTTD) and a *fingerprinting* method to determine the break points, e.g. using rolling checksum algorithms such as Adler-32 or Rabin.
- ...

Details on next slide

Deduplication & Versioning

> Syncany uses fixed-size chunking, but should use variable-size chunking

Fixed-Size Chunking

A:

5aaa	b2df	f612	81a4	71ad	ff7d
5aaa	a3df	f612	81a4	71ad	ff7d

*1 byte **replacement**, chunk boundaries remain*
5 out of 6 chunks are duplicates

B:

5aaa	b2df	f612	81a4	71ad	ff7d	
1f5a	aab2	dfd6	1281	a471	adff	7d

*1 byte **insertion**, all chunk boundaries shifted, no duplicates!*

Variable-Size Chunking

A:

5aaab2	dff6	f61	281	a471a	dff7d
5aaaa3	dff6	f61	281	a471a	dff7d

*1 byte **replacement**, chunk boundaries remain*
5 out of 6 chunks are duplicates

B:

5aaab2	dff6	f61	281	a471a	dff7d
1f5aaab2	dff6	f61	281	a471a	dff7d

*1 byte **insertion**, chunk boundaries remain*
5 out of 6 chunks are duplicates

uses

Sliding window based rolling hash

Chunk break condition:

***f* mod *D* = 0**

f is hash value
D is divisor

… f 6 1 2 8 1 a 4 …

Hash *(here: D = 0xff)*

f 6 1 2 8 → 0x22 mod 0xff = 0

6 1 2 8 1 → 0xff mod 0xff = 0

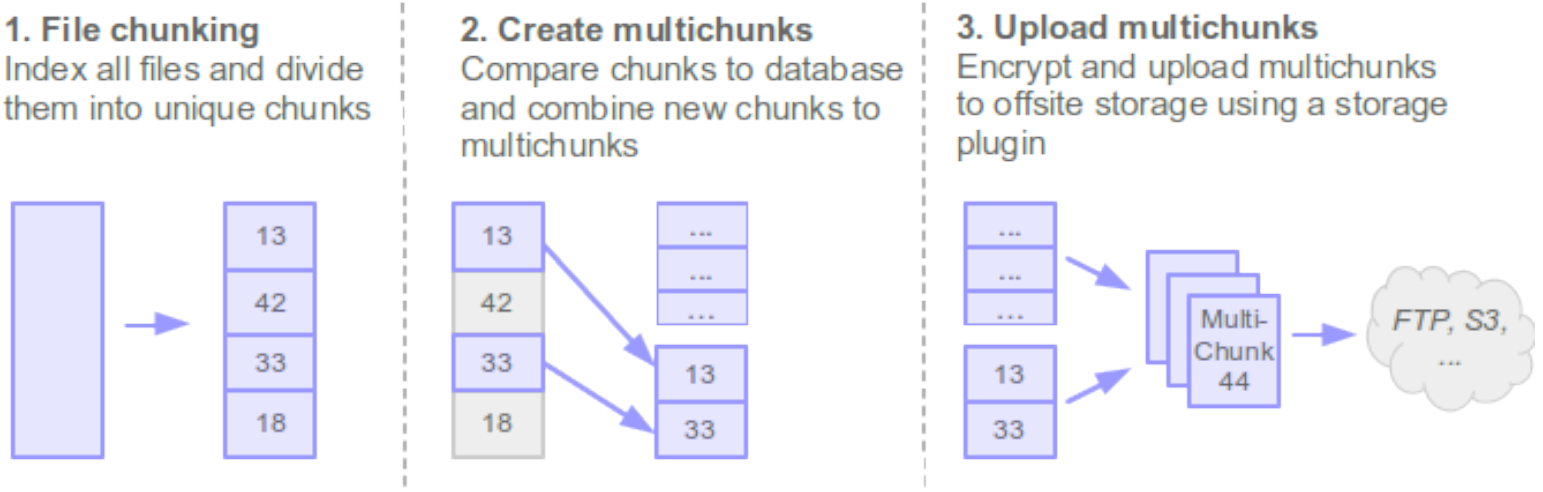
1 2 8 1 a → 0xf4 mod 0xff = 0

Legend:

- 5aaa — *New chunk*
- 5aaa — *Duplicate chunk*

Deduplication & Versioning

> Syncany: Fixed-size, client-side dedup. with additional multichunking

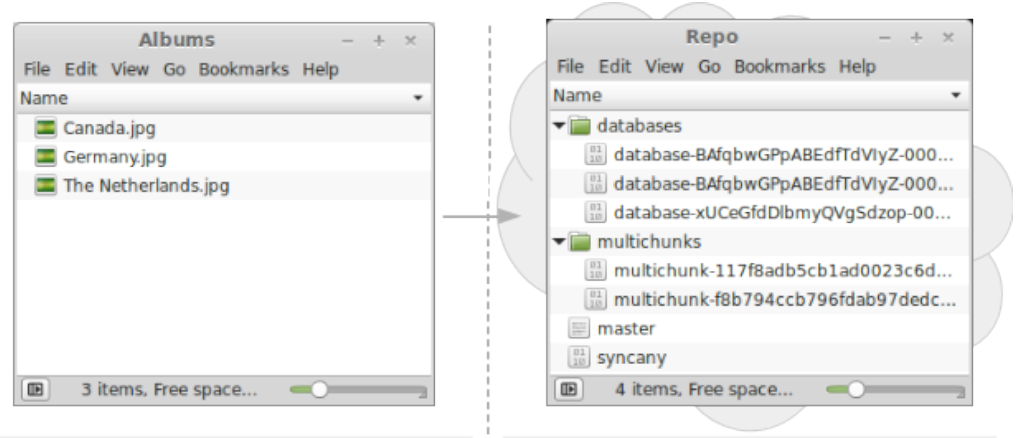


Current implementation:

- Fixed-size chunking, 512 KB chunks
- 4 MB multichunks

Also implemented, but not active:

- Variable-size chunking with TTTD algorithm, via Adler-32 fingerprinting



Local Syncany Folder

Syncany's local files are not encrypted, but are encrypted before upload.

Offsite Repository

Syncany stores file data in multichunks, and metadata in database files

Deduplication & Versioning

> Dedup. makes versioning easy, Syncany just keeps track of chunks

File 1a8a8dfeea4

		Chunks						
Presentation.ppt *(version 1)*	**NEW**	5aa..	b2d..	f61..	81a..	71a..	ff7..	+ 3.1 MB
Presentation.ppt *(version 2)*	**CHANGED**	5aa..	88a..	f61..	81a..	71a..	ff7..	+ 512 KB
Syncany.ppt *(version 3)*	**MOVED**	5aa..	88a..	f61..	81a..	71a..	ff7..	+/- 0 Bytes
Syncany.ppt *(version 4)*	**DELETED**							+/- 0 Bytes

File dfa818e192ad

		Chunks						
Products.ppt *(version 1)*	**NEW**	5aa..	b2d..	9ae..	e78..	aa1..	79e..	+ 2.1 MB

Free copies (copies of files add 0 byte to the repo)
Small file changes add minimal additional repo space
Restoring old/deleted files is easy

Agenda

- What is Syncany?
- How does it work?
- Focus Topics:
  - Deduplication & Versioning
  - **Cryptography Concept**
  - Continuous Integration & Packaging

Cryptography Concept
> Assumptions and Algorithms

Security Assumptions

- Storage provider and network infrastructure cannot be trusted
- The local machine is secure
- Users sharing a repository trust each other completely

Security Concepts

- Pre-upload encryption ← **Focus (details on next slide!)**
- HTTPS-only daemon/server

Algorithms:

- ***PBKDF2 with HMAC-SHA1, 1mm rounds, 512-bit salt***
  Password-Based Key Derivation Function 2, part of PKCS #5 v2.0
  Applies SHA1 multiple times on the password and salt to derive a key from the password
  Used in Syncany: per-repository master key generation

```
SecretKeyFactory factory = SecretKeyFactory.getInstance("PBKDF2WithHmacSHA1");
KeySpec pbeKeySpec = new PBEKeySpec(password.toCharArray(), salt, 1000000, 512);
SecretKey masterKey = factory.generateSecret(pbeKeySpec);
```

- ***HKDF with SHA256, 96-bit salt***
  HMAC-based Extract-and-Expand Key Derivation Function (HKDF), RFC5869
  Creates a derived key from the master key (input key material) and a salt
  Used in Syncany: per-file key is derived from the master key

```
HKDFBytesGenerator hkdf = new HKDFBytesGenerator(new SHA256Digest());
hkdf.init(new HKDFParameters(inputKeyMaterial, inputSalt, "Syncany..."));
byte[] derivedKey = new byte[outputKeySize / 8];
hkdf.generateBytes(derivedKey, 0, derivedKey.length);
```

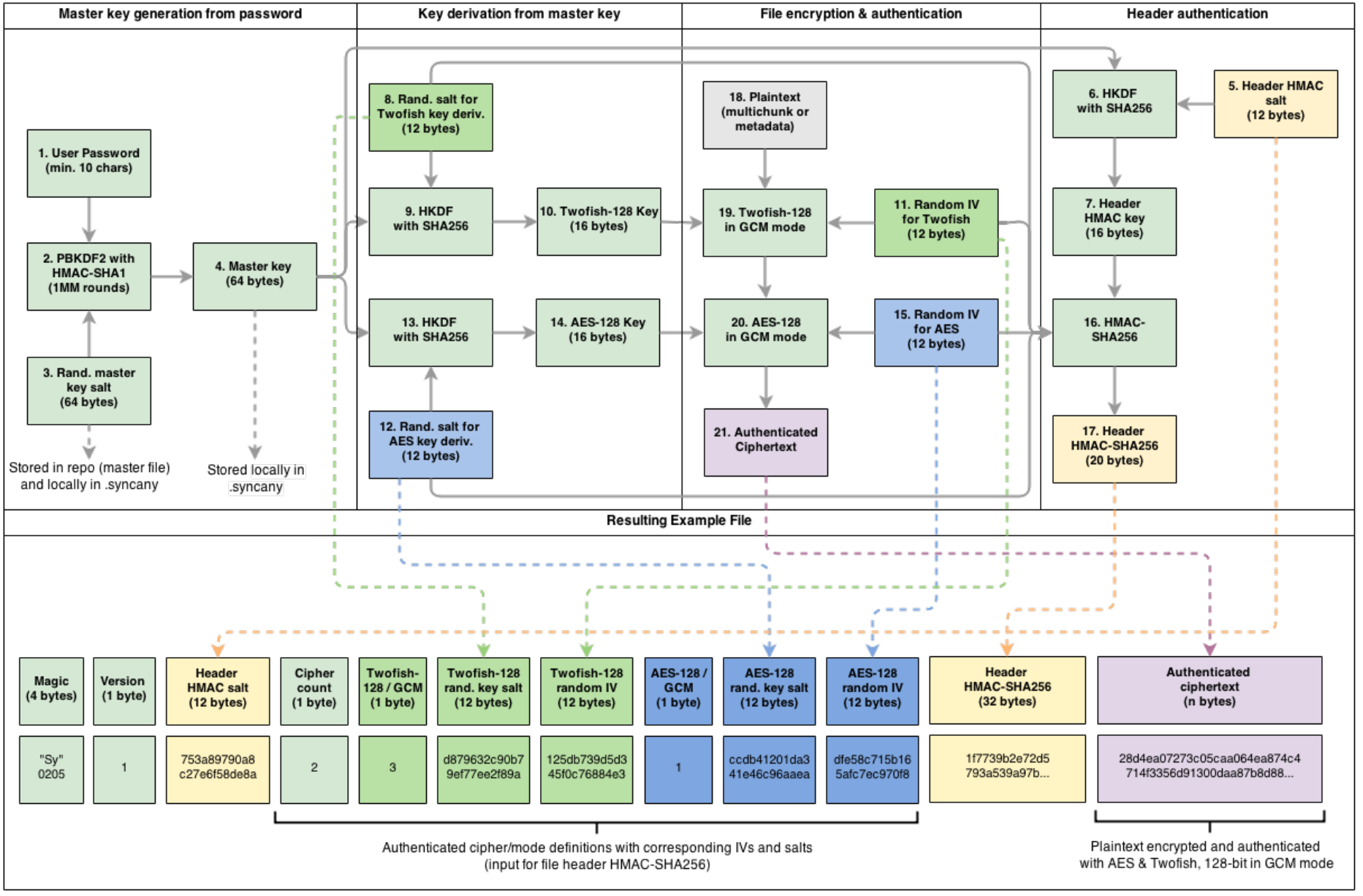
- ***HMAC-SHA256, key derived with HKDF, 96-bit salt***
  Keyed-Hash Message Authentication Code (HMAC), RFC2104
  Calculates a MAC using a cryptographic hash function (here: SHA256)
  Used in Syncany: per-file header authentication

```
headerHmac = Mac.getInstance("HmacSHA256", "BC");
headerHmac.init(hmacSecretKey); // via HKDF
headerHmac.update(headerBytes);
outputStream.write(headerHmac.doFinal());
```

- ***Twofish/AES 128-bit in GCM mode, key derived with HKDF, 96-bit salt***
  Symmetric encryption scheme, used in the Galois Counter Mode (GCM), an Authenticated Encryption with Associated Data (AEAD)
  Used in Syncany: per-file encryption; AES and Twofish in nested

Cryptography Concept

> Encryption file format and used algorithms



Agenda

- What is Syncany?
- How does it work?
- Focus Topics:
  - Deduplication & Versioning
  - Cryptography Concept
  - **Continuous Integration & Packaging**

Continuous Integration & Packaging

> Commit to `develop` to create a snapshot, or to `master` for a release

Workflow components:

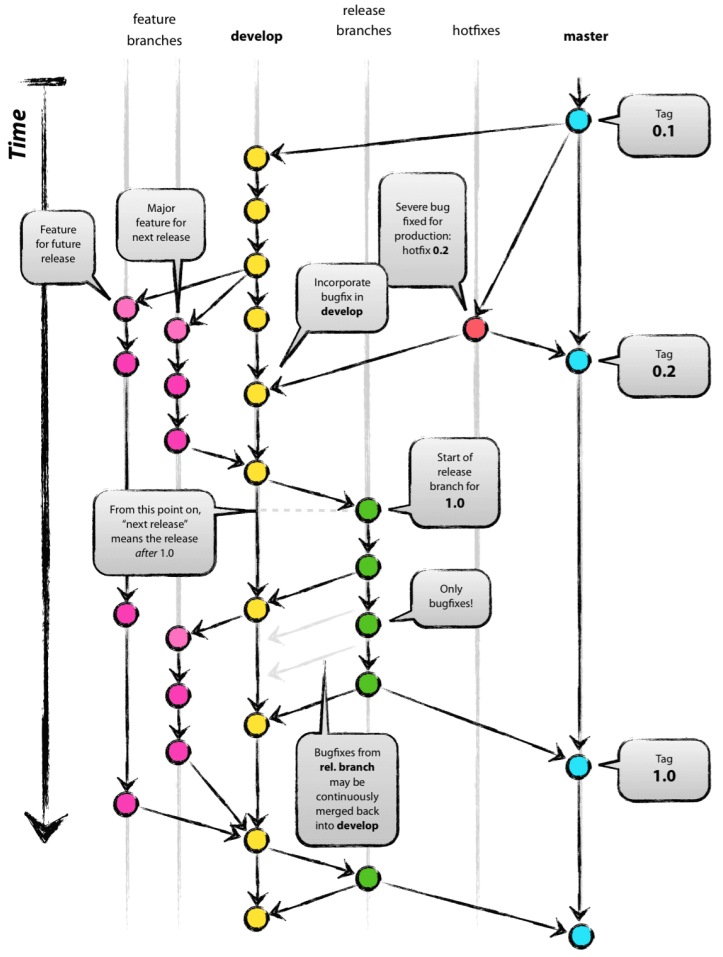
- Git Branching Model (as per nvie.com)
- Travis-CI (see travis-ci.org)

Branching model:

- Development happens on the **`develop`** branch
- The **`master`** branch is **only used for releases**, it is always production-ready
- Releases on the **`master`** are tagged with **vX.Y.Z** (semver)
- Branch naming:
  - **`feature/`** - Feature branches are prefixed (e.g. feature/issue333-history-browser)
  - **`hotfix/`** - Hotfix branches (e.g. feature/issue333-history-browser)

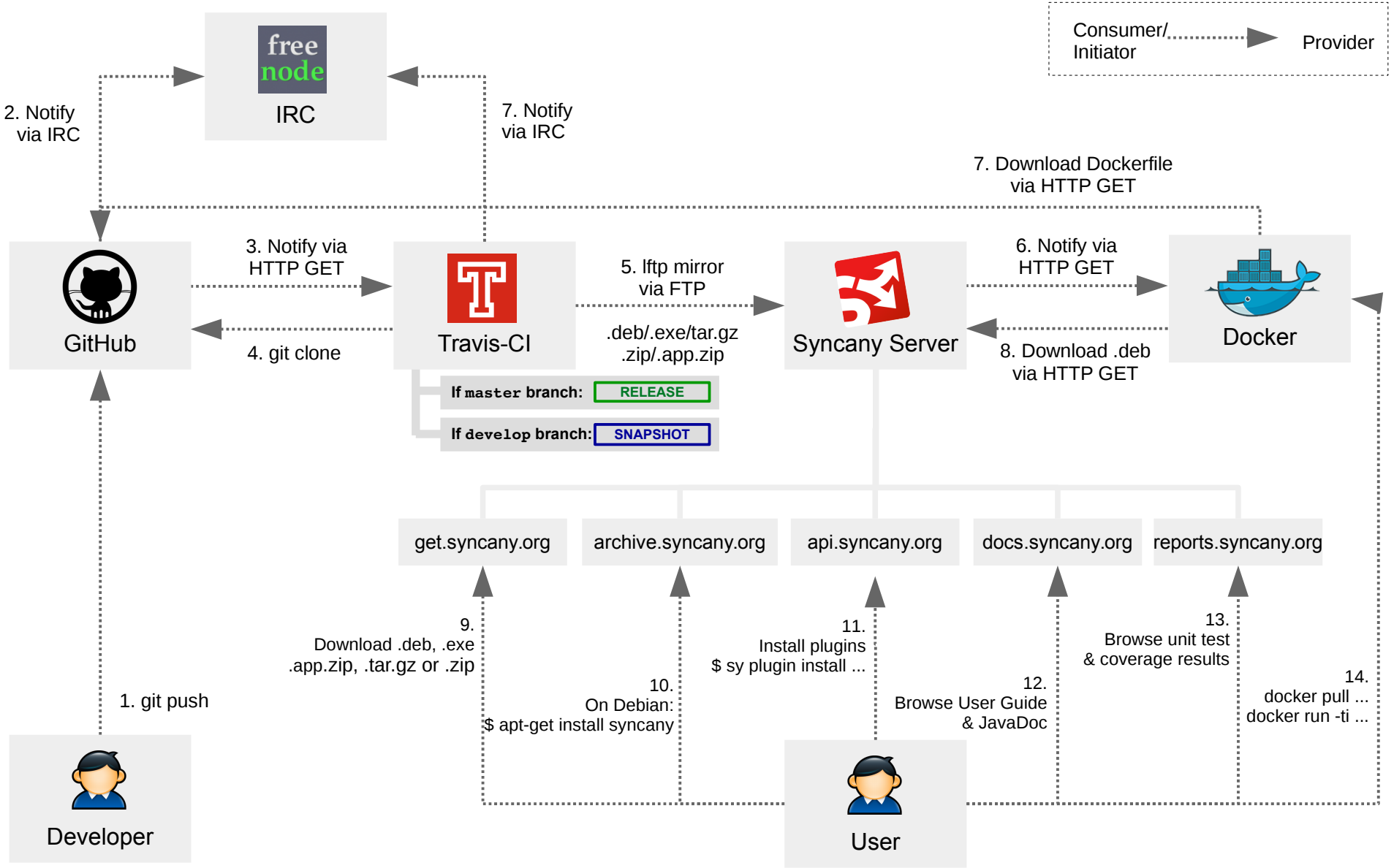
Travis CI

- Reacts to every commit
- Runs tests & builds packages, builds releases & snapshots



Continuous Integration & Packaging

> Things that happen when a developer runs “git push”



Continuous Integration & Packaging

> Debian/Windows/OSX packaging

All of the following steps are done with **every commit** to the **`develop`** or **`master`** branch. The result is either a snapshot or a new release.

Debian packaging & APT archive

- Travis: Automatic **`.deb`**-file packaging using **`debuild`**, and FTP upload to Syncany server
- Syncany server: Add to release/snapshot APT archive via **`reprepro`**
- Users: Download/update Syncany & plugins via **`apt-get install`**

Windows packaging

- Travis: Automatic creation of **Inno Setup** Windows installer (**.exe**), and FTP upload to Syncany server. *Speciality: The Inno Setup compiler is run inside Wine on Linux.*
- Syncany server: Place in release/snapshot download folder; Users: Download via website

OSX packaging

- Travis: Automatic creation of **.app.zip**-file
- Syncany server: Place in release/snapshot download folder; Users: Download via website

Other packages:

- Arch Linux PKGBUILD
- OSX homebrew recipe
- Docker image
- .zip and .tar.gz archives

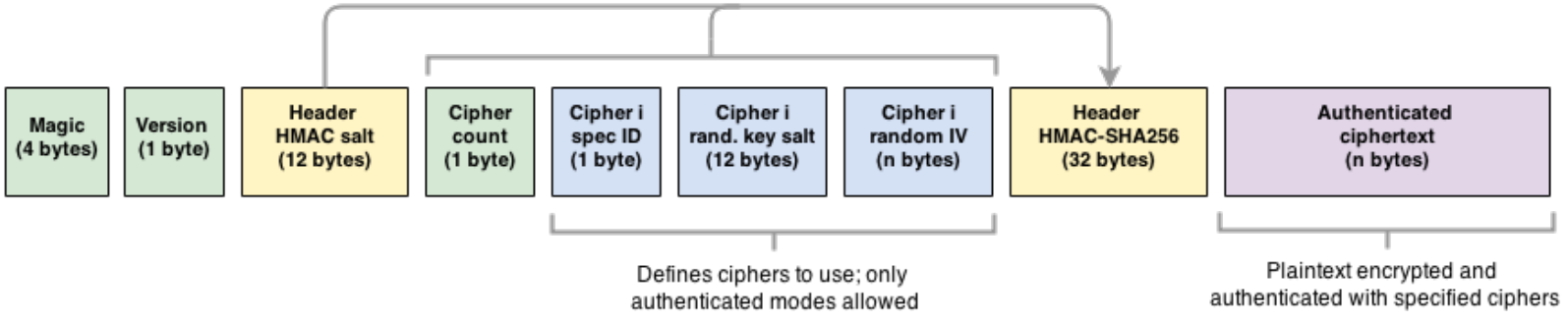
Links & Resources

- Website - https://www.syncany.org/
- User Guide - https://www.syncany.org/r/guide
- Code & Issues - https://github.com/syncany/syncany
- Wiki - https://github.com/syncany/syncany/wiki
- IRC - #syncany on Freenode - https://webchat.freenode.net/?channels=syncany
  (active, 20-30 people hang around here; 5-6 actually talk, my nick is “binwiederhier”)
- Videos on YouTube - https://www.youtube.com/channel/UCzegH3dpTK5HHQx6jJ5yhdQ
- Videos on Asciinema - https://asciinema.org/~binwiederhier

Backup

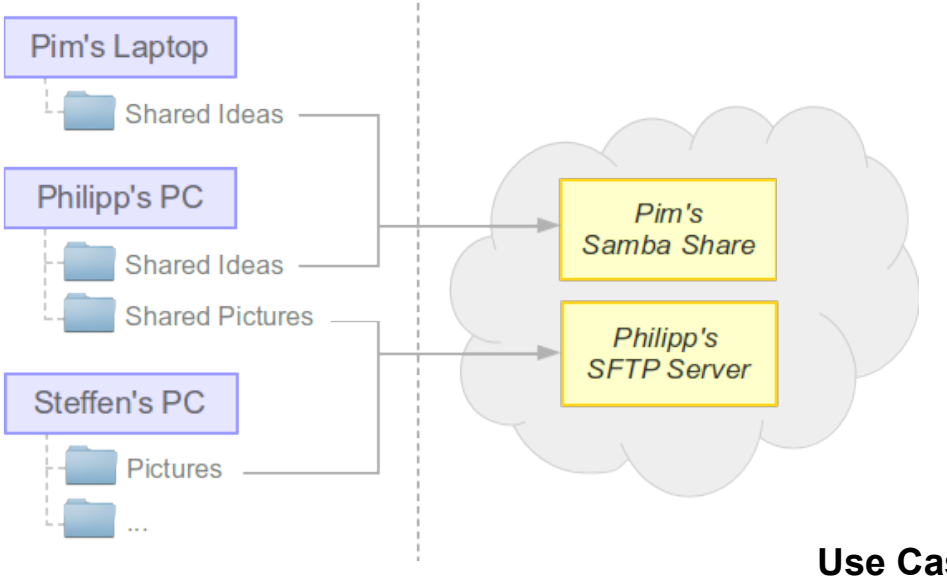
Cryptography Concept

> Generic File Structure

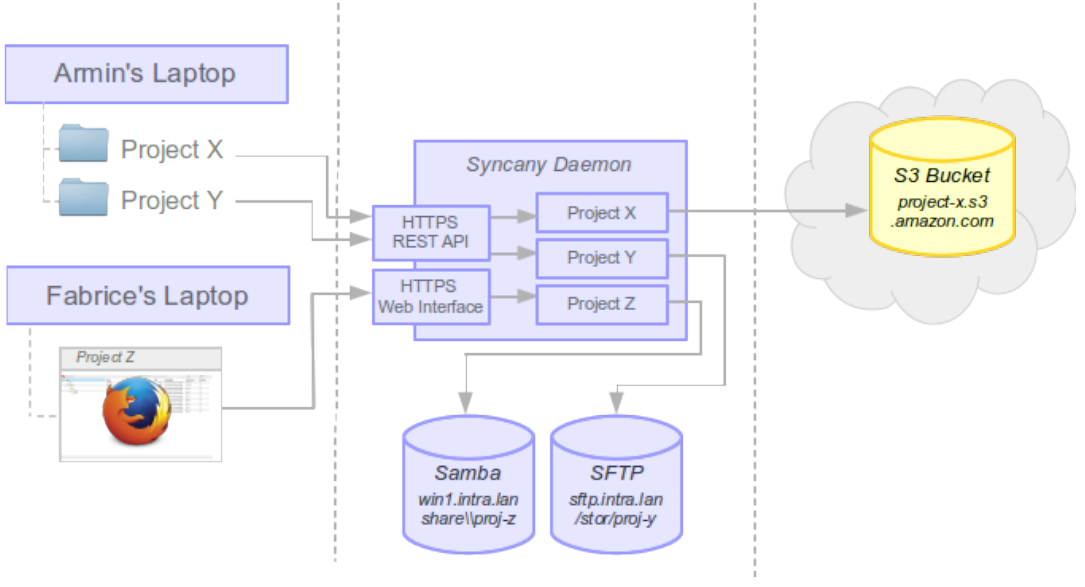


Use Cases

Use Case 1: Friends sharing files

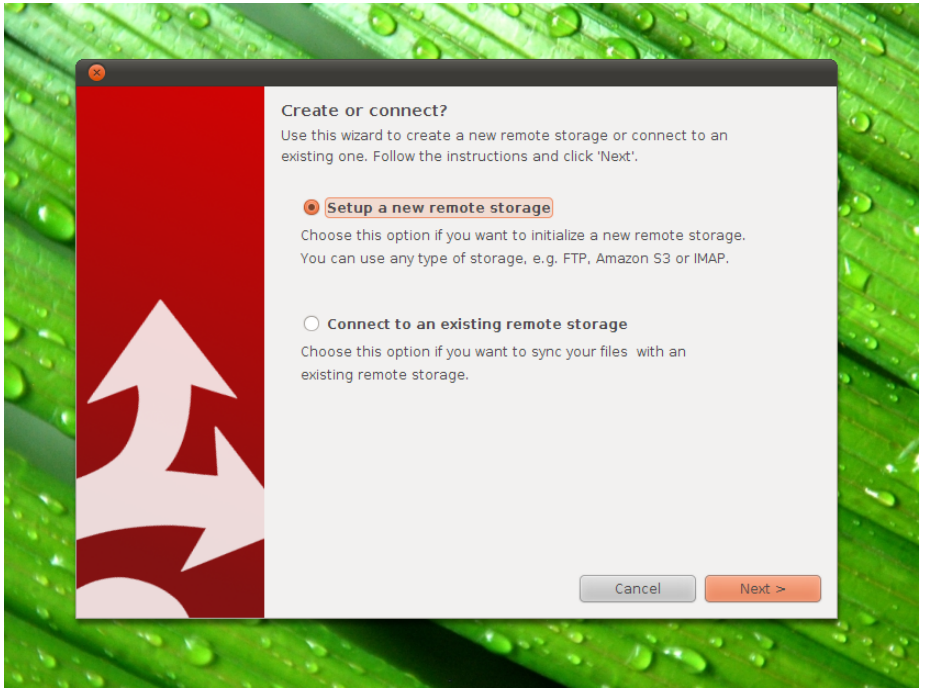


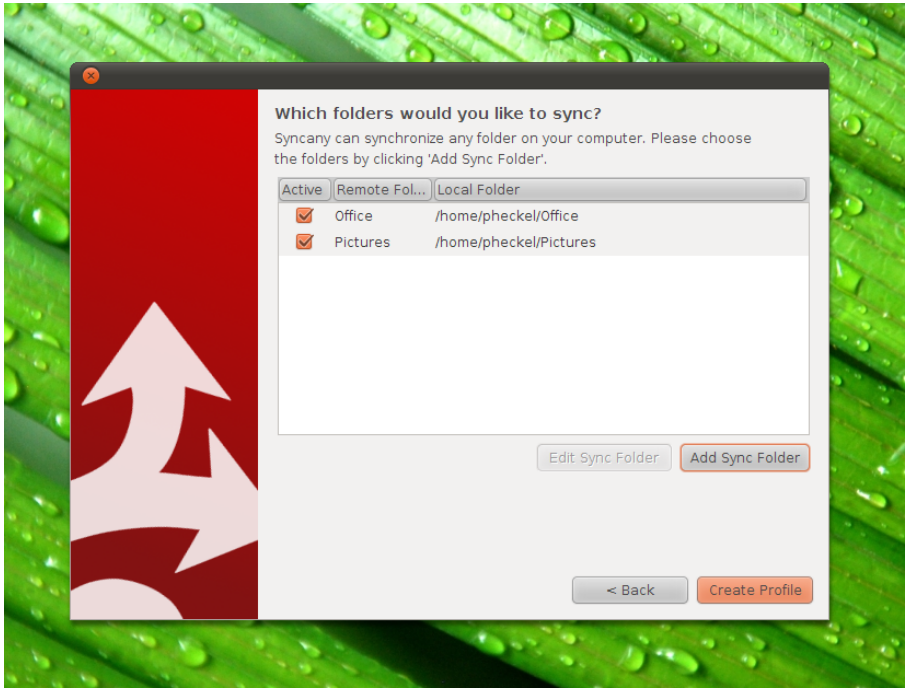
Use Case 2: Sharing in a company



Syncany Desktop User Interface

> Wizard Style Setup

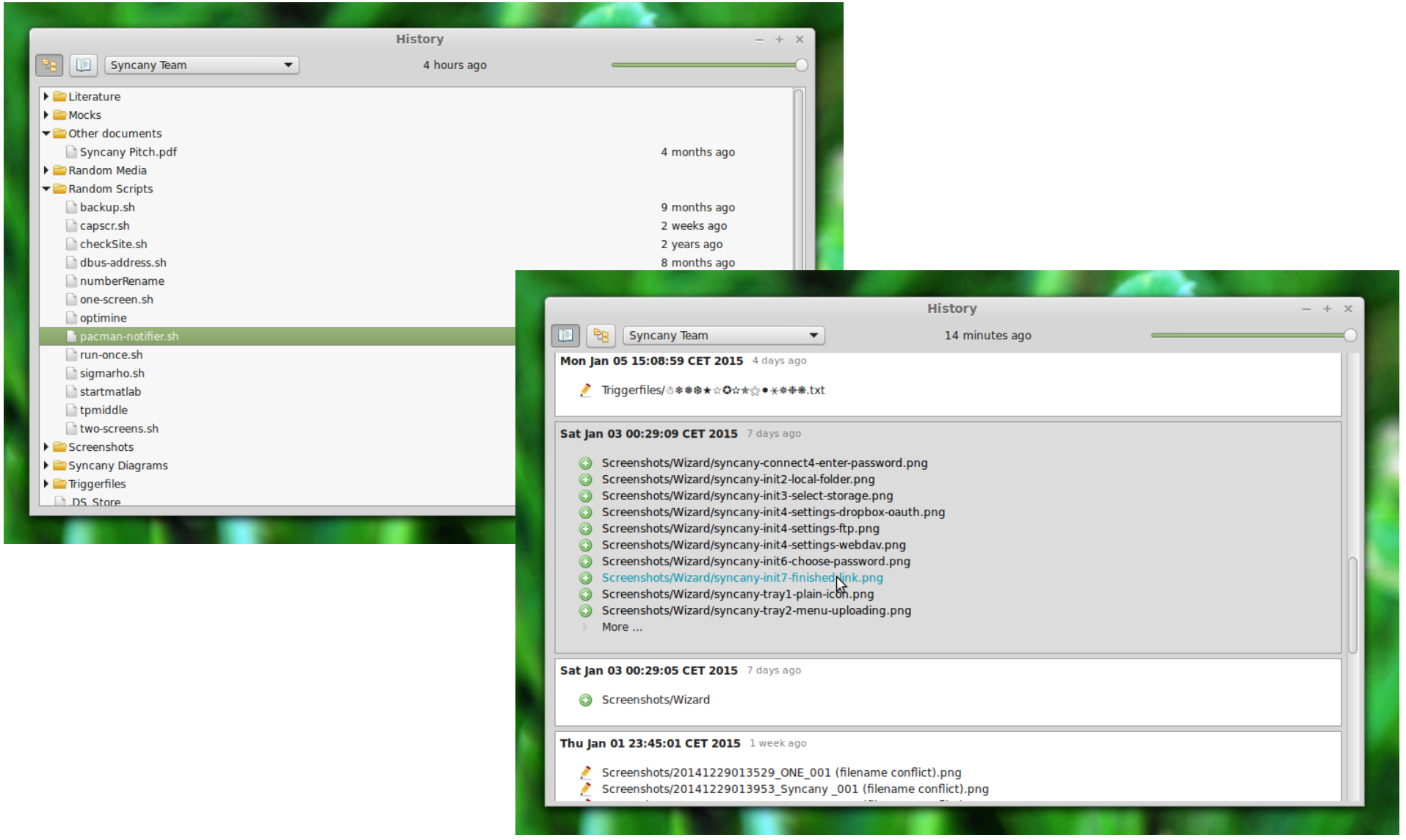




Cross-Platform Desktop Demo
https://www.youtube.com/watch?v=eHoA5_8gRBc

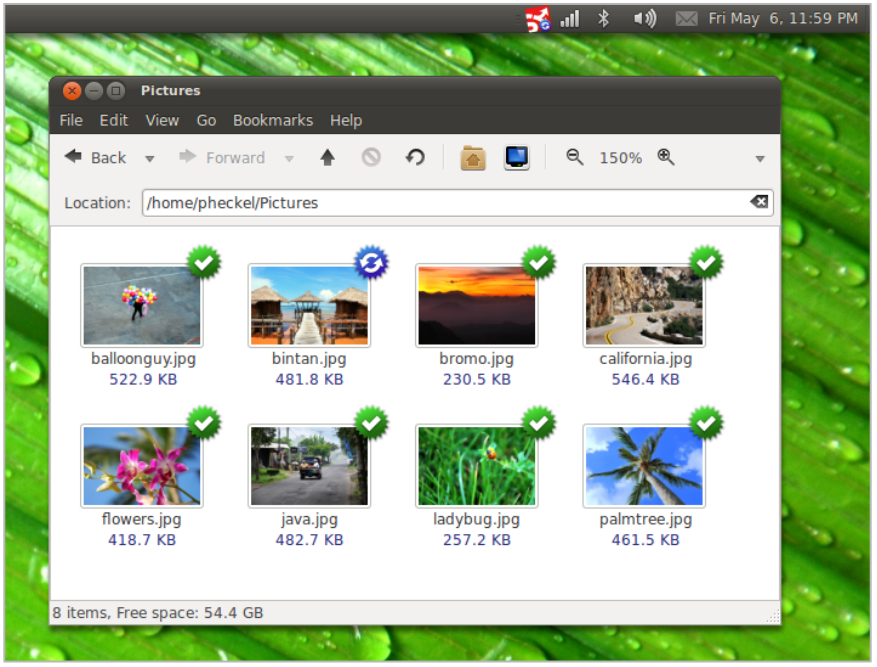
Syncany Desktop User Interface

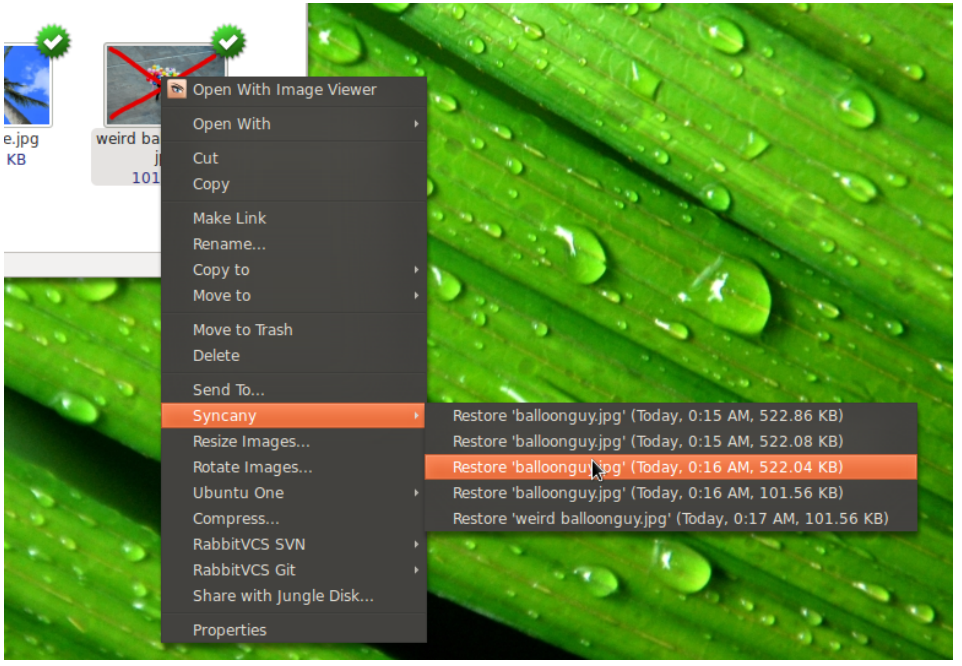
> History browser: Browse and restore old file versions



Syncany Desktop User Interface

> File Manager Integration (2011-version PoC, looking for devs)

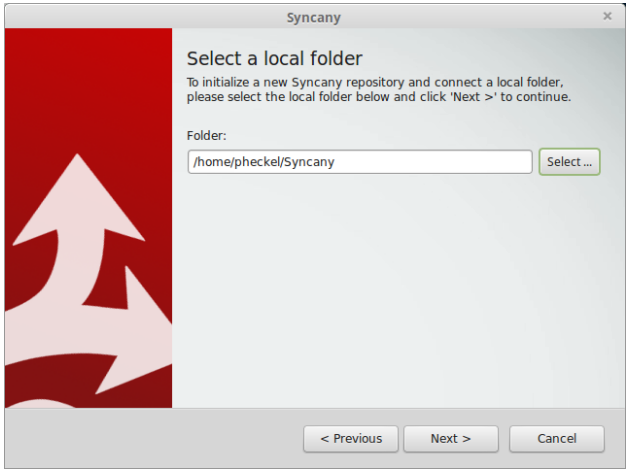




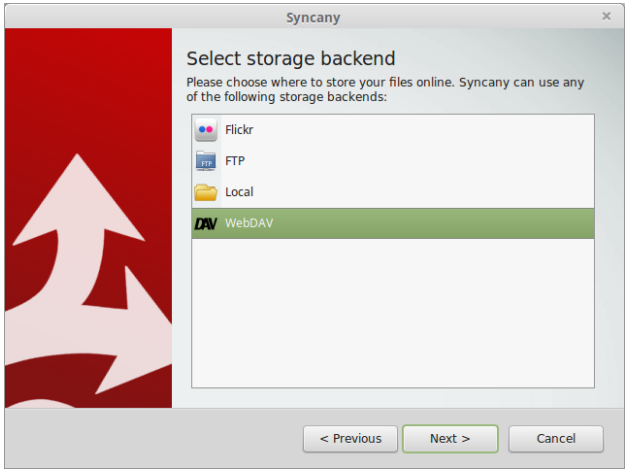
Cross-Platform Desktop Demo
https://www.youtube.com/watch?v=eHoA5_8gRBc

Using Syncany

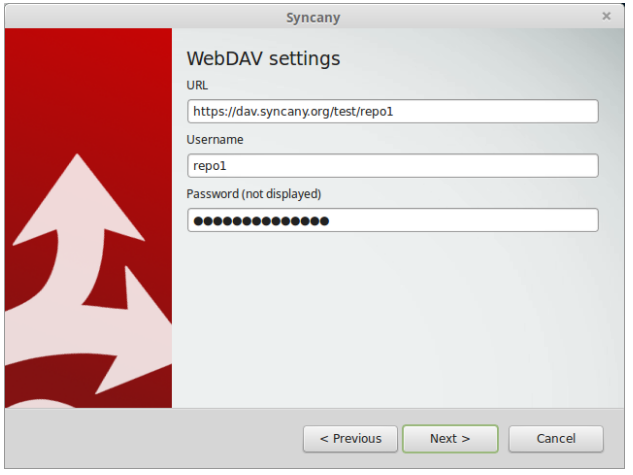
> Example: Syncing a local folder to a new WebDAV-based repository



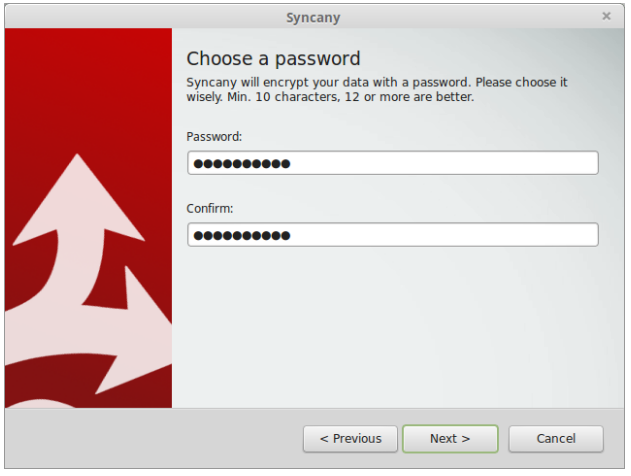
1. Choose the folder you want to sync



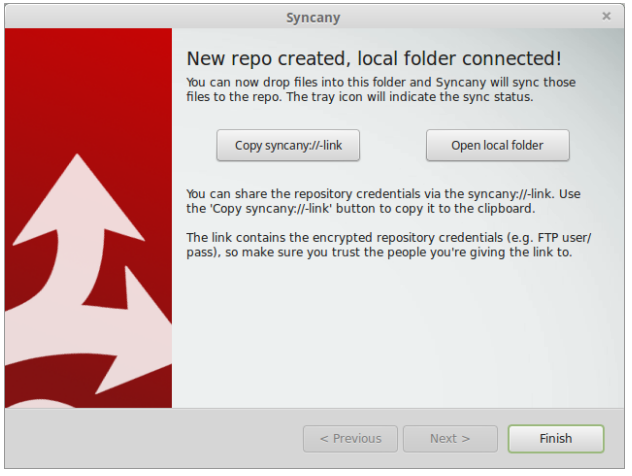
2. Choose a storage backend



3. Enter storage backend credentials



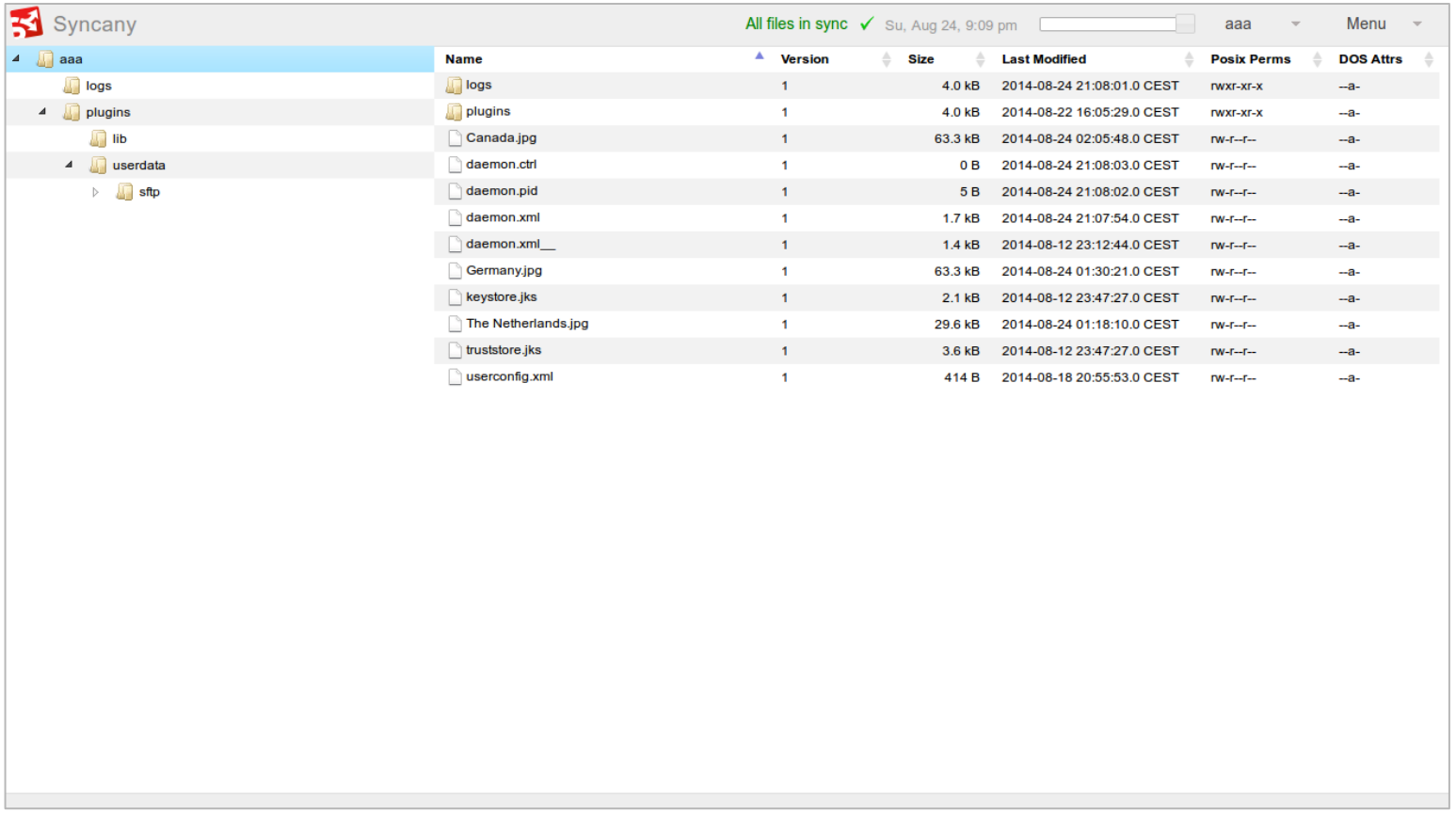
4. Choose password (used for encrypt.)



5. Done. Share a syncany://-link if you want.

Syncany Web Interface

> Web-based File Manager (proof-of-concept; looking for devs)



Web Interface Demo
https://www.youtube.com/watch?v=2NPVffywZVs