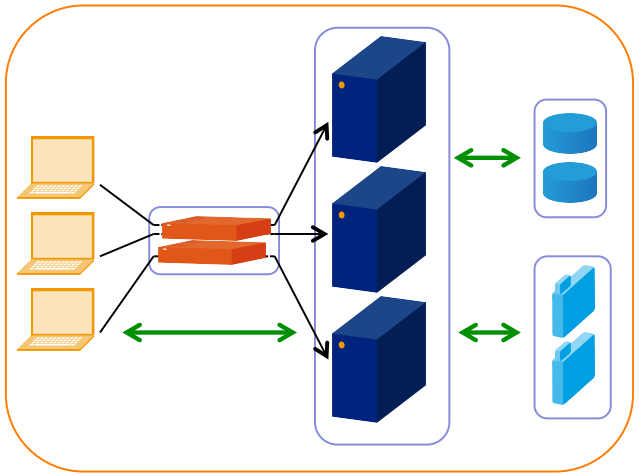
IdP Clustering

High Availability and Load Balancing



SWITCH

SWITCHaai Team
aai@switch.ch

You want to prevent

- HW failures
  - Server component failure
  - Power failure
  - Network failure
- Service overload
- Downtimes due to maintenance (major upgrades)
- …

What you usually do

- Take one box
  - Harden it through redundant components (power, network, disk, memory, CPU's, backplane (?))
- Or take another box
  - Organize failover (cold standby)
- Or take a couple of boxes
  - Organize load balancing
- Or take a VM from a "HA environment"

An academic question: Stateful or not?

- The IdP is stateful, simply because it maintains a “conversational state” with each of the clients.
- This conversational state is implemented in software other than the IdP, namely in Spring Web Flow, typically using a session ID.
- However, at present, there is no solution provided to replicate the per-request conversational state.
- So, this is a hard problem, and we have no solution out of the box. What can we do?

Storage Recommendations

Storage Entity	Recommended Storage	Scope
Persistent ID	***Common Database***	**Cluster**
User consent	***Common Database***	**Cluster**
IdP User Session	**Client**	**Per Client**
Transient ID (Backchannel)	*Common Database*	Cluster
SAML artifact	*Common Database*	Cluster
Conversation Session	Memory	Per Node
Message replay cache	Memory	Per Node

Remarks:

- https://wiki.shibboleth.net/confluence/display/IDP30/Clustering
- "Common Database" means some central/clustered database or a database replicated between nodes.
- SAML artifact:
  Irrelevant if SAML 2.0 artifacts not used/required at all
- Alternatives for Message replay cache:
  Common Database or memcached (depending on security requirements)

From the IdPv3 business case

- "The choice of **Terracotta** as a primary clustering solution for high availability has not worked out particularly well for the project and we have been evaluating possible directions and design implications from the early planning stages. While the original intent was to move toward a technology called **Infinispan** as a replacement, recent experience from the community has not been positive (feedback for which we are tremendously appreciative)."

+ "Much design attention has been given to ensuring support will be possible for other popular solutions such as **databases** and **memcached**."

https://shibboleth.net/documents/business-case.pdf

Do it yourself

Then you need to think about

- Network
- Processing (CPU, memory)
- Persistent storage (Disks and DBs)

Tools

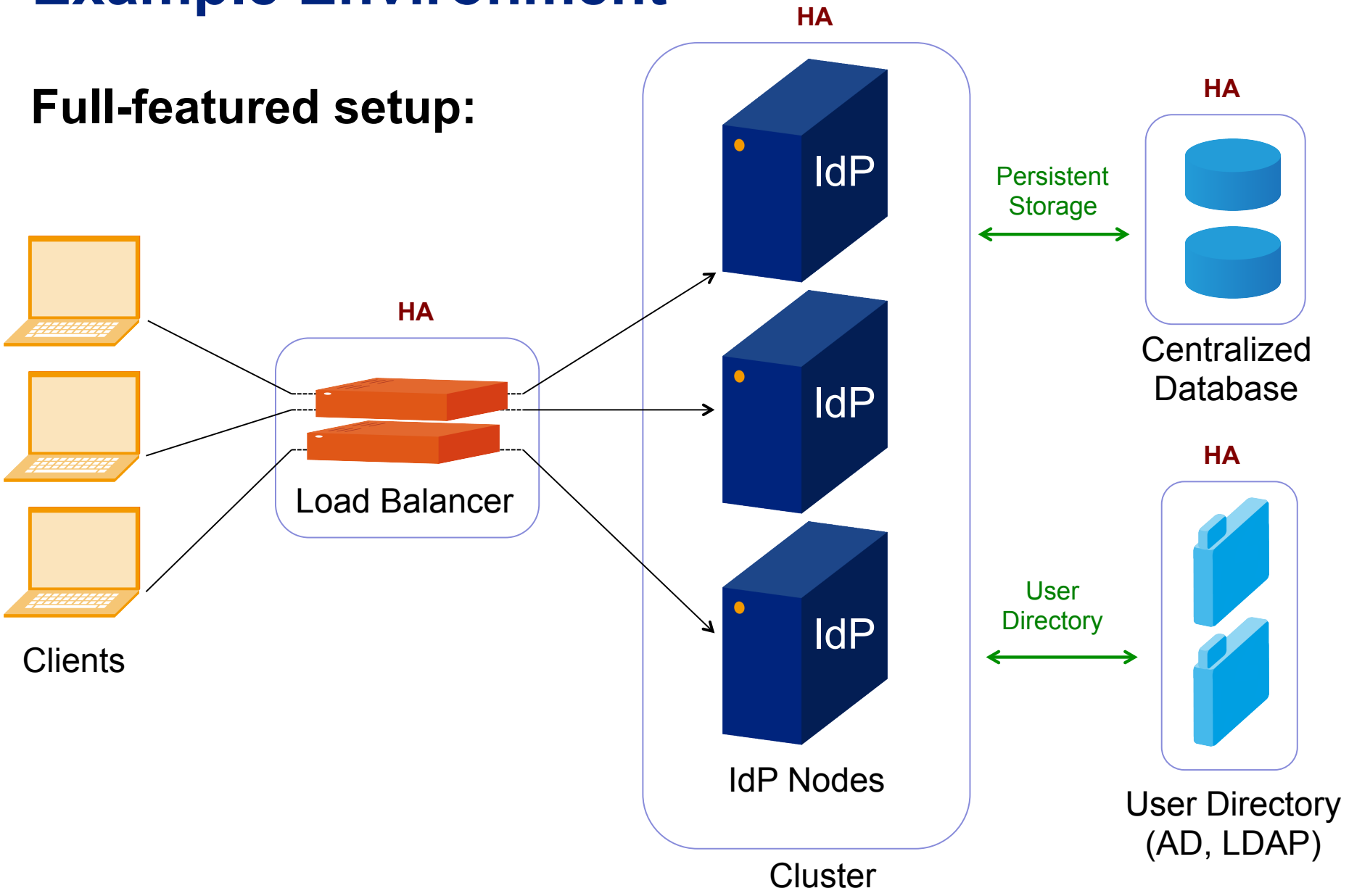
- NGINX: popular HTTP load balancer, but additional features may cost
- PostgresDB: recommended DB by Shibboleth
- Memcached: Cache or even alternative to DBs, in-memory, key-value data store.
- DRDB: a “network based raid-1 block devices to put a filesystem on”

Examples

Who	Network	Processing	Persistent storage
Uni Bern (IdPv3)	NGINX (active-active) HTTP Loadbalancer	2 IdPs	Use of central MSSQL-cluster
Uni Genève (IdPv2)	F5 BIG-IP Loadbalancer (sticky)		MySQL DB Cluster
Uni Lausanne (IdPv2)	HW load balancer (active-passive)	2 IdPs (active-passive)	external MySQL-DB (also HA: Heartbeat + DRBD)
Uni Zürich (IdPv2)		3 IdPs	external MySQL database
HES-SO Fr (IdPv2)		2 IdPs active-active	
Uni Marburg (IdPv2)	NGINX Loadbalancer	2 IdPs, memcached,	1 external PostgresDB server
SWITCH (IdPv2)	Anycast address	2 IdPs active-passive	Local MySQL-DB, replicated by cron

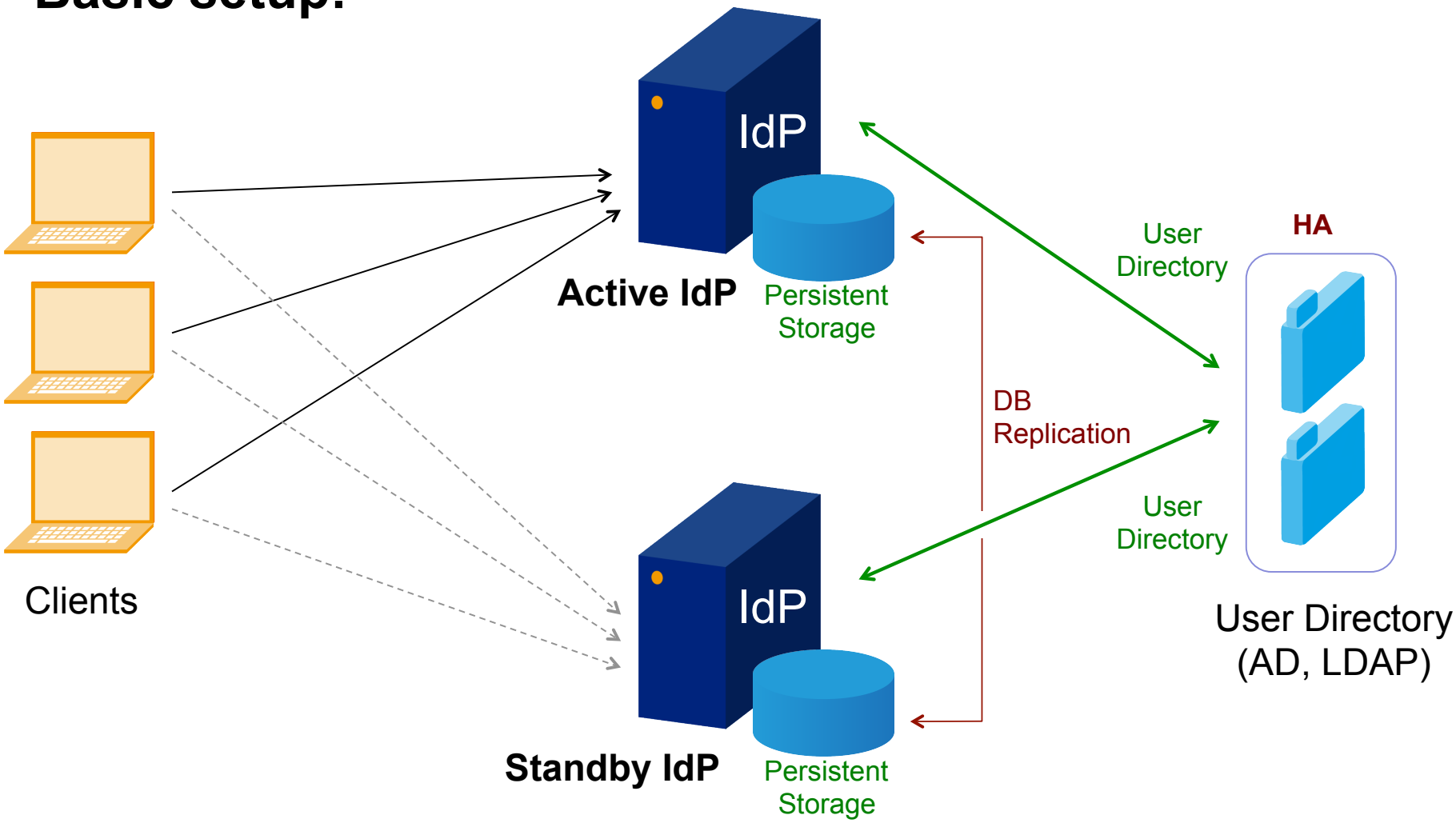
Example Environment

Full-featured setup:



Example Environment

Basic setup:



References

Documentation

- **Clustering**
  https://wiki.shibboleth.net/confluence/display/IDP30/Clustering
- **Secret Key Management**
  https://wiki.shibboleth.net/confluence/display/IDP30/SecretKeyManagement
- **Storage**
  https://wiki.shibboleth.net/confluence/display/IDP30/Storage
- **Discussion on Persistence**
  https://wiki.shibboleth.net/confluence/display/IDP30/Persistence

Thanks to

- **Dominique Petitpierre**
- **Manuel Haim**
- **Michael Pfister**
- **Daniel Lutz**
- **Lukas Hämmerle**
- **Many others**