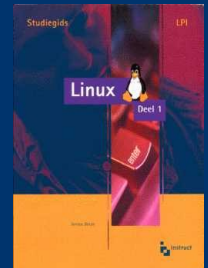
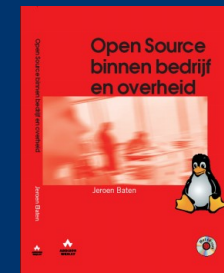
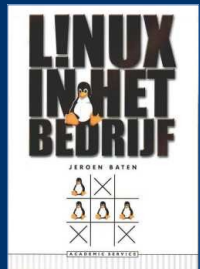


Making Ansible playbooks to configure Single-Sign-On for popular open source applications

Who am I?

- Jeroen Baten (English/Spanish: Yerun)
- Job title : Open Source expert @ Chateau IT
- Author of 12 books (4 more in beta)
- Dad of 5 girls
- (former) volunteer fire fighter
- Scouting
- Trainer, teacher, hacker



What do I do?

- Open source consultancy
- Teaching/training
(Python, Web, Linux, Zabbix, etc)
- Chateau IT: job retraining, from X to IT
 - **Whatever your background, if you want to switch to a career in IT, contact me jeroen@chateau-it.nl**



Another project



CHÂTEAU IT

IT OPLEIDING IN FRANKRIJK

- LibrePlan **LIBREPLAN**
- Web-based project management appl.
- Very cool!
- www.LibrePlan.dev

The screenshot displays the LibrePlan web application interface. The top navigation bar includes tabs for 'Scheduling', 'Resources', 'Administration / Management', 'Reports', and 'My account'. The main content area shows a 'WBS (tasks)' table with columns for 'Scheduling state', 'Code', 'Name', 'Hours', 'Must start after', and 'Deadline'. The table lists various project tasks, including 'Project Coordination', 'Graphical design', 'Initial Drafts', 'Static prototypes', 'Brand deliverables', 'HTML/CSS templates', 'Feature A implementation', 'Feature B implementation', 'Subfeature B.1', 'Subfeature B.2', 'Add functional Testing', and 'Quality Assurance'. A sidebar on the left contains icons for 'Project Scheduling', 'Project Details', 'Resources Load', 'Advanced Allocation', and 'MonteCarlo Method'.

Scheduling state	Code	Name	Hours	Must start after	Deadline
		Project Coordination	200		
		Graphical design	140		
		Initial Drafts	60		
		Static prototypes	40		10/12/11
		Brand deliverables	20		
		HTML/CSS templates	80		
		Feature A implementation	70		
		Feature B implementation	150		
		Subfeature B.1	40		
		Subfeature B.2	110		
		Add functional Testing	60		
		Quality Assurance	20		

Let me start with apologies first...



CHÂTEAU IT

IT OPLEIDING IN FRANKRIJK

Talking the same language... can still cause a culture clash.
With respect to the code of conduct:
Please forgive me where applicable.
(Tip: Dutch people take every English text literally!)

ANGLO-DUTCH TRANSLATION GUIDE		
WHAT THE BRITISH SAY	WHAT THE BRITISH MEAN	WHAT THE DUTCH UNDERSTAND
<i>With all due respect ...</i>	<i>I think you are wrong.</i>	<i>He is listening to me.</i>
<i>Perhaps you would think about ... I would suggest ...</i>	<i>This is an order. Do it or be prepared to justify yourself.</i>	<i>Think about this idea and do it if you like.</i>
<i>Oh, by the way ...</i>	<i>The following criticism of the purpose of the discussion is ...</i>	<i>This is not very important.</i>
<i>I was a bit disappointed that ...</i>	<i>I am very upset and angry that ...</i>	<i>It doesn't really matter.</i>
<i>Very interesting ...</i>	<i>I don't like it.</i>	<i>They are impressed.</i>
<i>Could you consider some other options?</i>	<i>Your idea is not a good one.</i>	<i>They have not yet decided.</i>
<i>Please think about that some more.</i>	<i>It's a bad idea. Don't do it.</i>	<i>It's a good idea. Keep developing it.</i>
<i>I'm sure it's my fault.</i>	<i>It's not my fault.</i>	<i>It was their fault.</i>
<i>That is an original point of view.</i>	<i>Your idea is stupid.</i>	<i>They like my ideas!</i>

The project in short



- Build IT landscape @ company and copied for other company in the group.
- Foundation: Proxmox, FreeIPA LDAP
- Installed applications Xwiki, Zabbix, Jenkins, Nextcloud, GitLab, Odoo, CMDBuild etc in separate vm's.
- Got question how to upgrade the landscape.
- So I proposed to make everything SSO using Ansible playbooks.
- And so our adventure started...

Basic Lingo



- Application that uses SSO = SP
 - Service (because application) provider
- Application that does SSO = IdP
 - Identity provider, in our case: Keycloak
- ACS: Assertion Consumer Service URL (SP sign-in URL)
- Ansible
 - Language to write configuration recipes
- JSON
 - The only good thing that came from JavaScript :-)

Basic SSO process flow



- User clicks 'login' on some application (SP)
- Browser of user is redirected to IdP (Keycloak)
- User is presented with login widget
- User logs in (successfully) or error/denied message.
- If not yet 2FA configured but set as mandatory he/she gets 2FA setup dialog.
- Browser of user is redirected to SP with some credentials proven he has successfully logged in at IdP.
- User is logged in.
- Every other application login redirects to IdP.
- IdP sees existing ticket of user and redirects immediately with authentication info.

Basic SSO setup



- User-id's in FreeIPA
- Keycloak for web SSO server, syncs with FreeIPA
- Keycloak has a client definition for every connected application
- Added first application (Xwiki)
 - This was a walk in the park, good documentation.
- Added another application, etc.

Keycloak clients list

The screenshot shows the Keycloak Admin Console interface. On the left is a dark sidebar with a menu containing 'Manage' (Clients, Client scopes, Realm roles, Users, Groups, Sessions, Events) and 'Configure' (Realm settings, Authentication, Identity providers, User federation). The main content area is titled 'Clients' and includes a description: 'Clients are applications and services that can request authentication of a user. Learn more'. There are two tabs: 'Clients list' (active) and 'Initial access token'. Below the tabs is a search bar with the text 'Search for client', a 'Create client' button, and an 'Import client' link. A table lists the clients with columns for Client ID, Type, Description, and Home URL. The table contains 8 rows of client data. At the bottom right of the table, there is a pagination control showing '1-8' and navigation arrows.

Client ID	Type	Description	Home URL
account	OpenID Connect	-	https://ad.onestein2.lan/realms/ONESTEIN2.LAN/account/
account-console	OpenID Connect	-	https://ad.onestein2.lan/realms/ONESTEIN2.LAN/account/
admin-cli	OpenID Connect	-	-
broker	OpenID Connect	-	-
https://cmdb-a.onestein2.lan/cmdbuild	SAML	-	https://cmdb-a.onestein2.lan/cmdbuild
https://cmdb.onestein2.lan/cmdbuild	SAML	-	https://cmdb.onestein2.lan/cmdbuild
realm-management	OpenID Connect	-	-
security-admin-console	OpenID Connect	-	https://ad.onestein2.lan/admin/ONESTEIN2.LAN/console/

- Let's have a look at the program flow

Ansible playbook flow



- Two Ansible variable files: Global-vars and encrypted-vars
- Playbook works on application vm
- Playbook retrieves Keycloak endpoint info
- Playbook checks if Keycloak client exists, if yes, deletes
- Playbook fills client definition template and uploads to Keycloak
- Checks if client created successfully
- Downloads shared secret if relevant (open-idc)
- Ansible leaves you with a configured application
- Displays remaining manual tasks, if any.

Ansible SAML example: Zabbix



- Read global vars, Read encrypted content
- Download Zabbix 5.4 repo package for Ubuntu 20.04, install 5.4 repo list,
- Install all needed packages
- Configure zabbix database password
- Setup Zabbix Postgresql database user, Setup Zabbix Postgresql database, Load initial Zabbix dataset when db just created
- Make SSL dir for nginx, Copy SSL key and cert to ssl dir, Install nginx config file
- Generate key on Zabbix server
- Retrieve token url from Keycloak server, Store url for easier retrieval, Retrieve endpoint info for our realm `{{ realm }}`, Store `authorization_endpoint` for faster retrieval, Store `token_endpoint` for faster retrieval
- Store `userinfo_endpoint` for faster retrieval, Retrieve authentication token from token-service url, Store access token into variable for easier retrieval
- Retrieve IDP metadata descriptor to use the 509 formatted certificate, Save IDP XML metadata to file for processing
- Run `xmlstarlet` to retrieve X509Certificate, Store output in certificate variable
- Create `idp.crt` file
- Retrieve current list of clients and search for already existing `"{{ zabbix_client_id }}"`
- Find ID in returned json
- copy remote ssl files to remote `/tmp`, Remove first line from tmp files, Retrieve remote ssl cert, Retrieve remote ssl key
- Delete client id `"{{ zabbix_client_id }}"` if it already exists.
- Convert Ninja template to variable
- Upload JSON template file to create new Client ID on Keycloak server
- If all went well we now have a location of the newly created Client ID
- (Re)start Zabbix server
- (Re)start Nginx server
- Post-install message IT IS IMPORTANT TO READ THIS

Ansible JSON tricks



- "baseUrl": "{{ zabbix_server_url }}",
- "adminUrl": "{{ zabbix_server_url }}/index_sso.php?acs", ← application specific Acs
- "saml_single_logout_service_url_redirect": "{{ zabbix_server_url }}/index_sso.php?sls", ← application specific
- "id": "{{ lookup('community.general.random_string', length=20) | to_uuid }}",
- "saml.signature.algorithm": "RSA_SHA256",
- "saml.signing.certificate": "{{ sp_cert.stdout }}",
- "saml.signing.private.key": "{{ sp_key.stdout }}",
- "saml_force_name_id_format": "true",
- "saml_name_id_format": "username", "user.attribute": "email",
- Template sent to Keycloak has random ID's
 - "protocolMappers": [{
 - "id": "{{ lookup('community.general.random_string', length=20) | to_uuid }}",
 - "name": "zabbixuser",
 - "protocol": "saml",
 - "protocolMapper": "saml-user-attribute-mapper",
 - "consentRequired": false,
 - "config": {
 - "user.attribute": "email",
 - "friendly.name": "email",
 - "attribute.name": "email"
 - "saml.multivalued.roles": "false",
 - "name": "role list",

Do-it-yourself (DIY)



- Once you have a working SSO setup:
 - Use `contrib/get-keycloak-client-list.sh`
 - Redirect to file
 - Cut out working client definition
 - Pipe through `jq` program
 - Start replacing settings with variables
 - Tool: `diff <(jq --sort-keys . $1) <(jq --sort-keys . $2)`

Gotchas



- Everything works better when using http**S**(!)
- Tomcat expects ssl keystore to have password 'changeit'
- Some application developers can't read. If the standard says 'optional' that is NOT the same as 'mandatory'.
- (Some/all) applications are very badly documented.
- Adding FreeIPA → Keycloak user-id sync midway was not a smart idea.
- Ansible can solve just about any problem
- Do NOT use Keycloak 18.x.y
 - unless you like long searches why roles don't work

Your job
(if you chose to accept
it)



CHÂTEAU IT

IT OPLEIDING IN FRANKRIJK



gl

Thank you for your attention!



Questions for me?: jeroen@chateau-it.nl