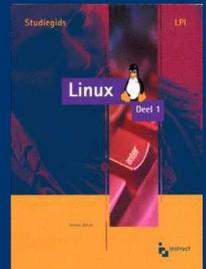
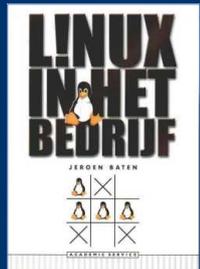


Making Ansible playbooks to configure Single-Sign-On for popular open source applications

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?



- Company: Chateau IT
- IT Consultancy and training
- Specialised in:
 - **Burnout Prevention Training**

Another project



CHÂTEAU IT

OPLEIDINGEN 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, with 'Subfeature B.1' highlighted in yellow. 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		

The project in short



- I build an IT landscape @ Onestein and copied for other company in the group.
- Onestein does Odoo ERP implementations and sponsors lots of open source projects.
- Infra foundation: Proxmox VM's, 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 everythi

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 called 'playbooks'
- JSON
 - The only good thing that came from JavaScript :-)

Basic SSO process flow



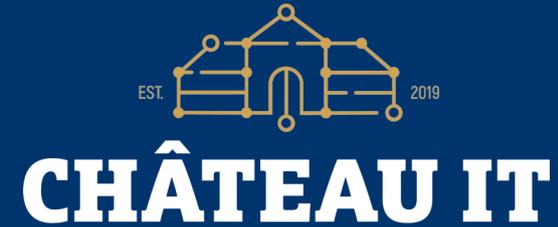
- User clicks 'login' on some application (SP)
- Browser of user is redirected to IdP (Keycloak)
- User is presented with login widget
- User logs in (successfully) or error/denied message.
- If not yet 2FA configured but set as mandatory he/she gets 2FA setup dialog.
- Browser of user is redirected to SP with some credentials proving 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 are held in in FreeIPA LDAP
- Keycloak for web SSO server, syncs with FreeIPA
- Keycloak has a “client” definition/config for every connected application
- Added first application (Xwiki) in 20 minutes
 - This was a walk in the park, good documentation.
- Added another application, etc.

Keycloak clients list



The screenshot shows the Keycloak Admin Console interface. At the top left is the Keycloak logo and a hamburger menu. The top right shows the user 'admin' and a profile icon. The left sidebar contains a navigation menu with options like 'Manage', 'Clients', 'Client scopes', 'Realm roles', 'Users', 'Groups', 'Sessions', 'Events', 'Configure', 'Realm settings', 'Authentication', 'Identity providers', and 'User federation'. The main content area is titled 'Clients' and includes a sub-header 'Clients are applications and services that can request authentication of a user. Learn more'. Below this are tabs for 'Clients list' and 'Initial access token'. A search bar is present with the text 'Search for client' and a 'Create client' button. 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.

- 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, delete!
- Playbook fills client definition template and uploads to Keycloak
- Checks if client conf is 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 server



- 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, Retrieve endpoint info for our realm `{{ realm }}`, Store `authorization_endpoint`, Store `token_endpoint`
- Store `userinfo_endpoint`, Retrieve authentication token from token-service url, Store access token into variable
- 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
- Display ost-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 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

OPLEIDINGEN IN FRANKRIJK



Your job (if you chose to accept it)



CHÂTEAU IT
OPLEIDINGEN IN FRANKRIJK

- Add applications and build this further!
- Git pull https://github.com/kwoot/Project_Single_Sign_On
- For SAML start with Zabbix playbook
- For OpenID Connect start with Xwiki playbook
- Every application is documented in the wiki
- Tweak and tune to a working confi

Thank you for your attention!



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