Automated sudo **usage control**  with **Ansible**

**1. Background**

1. Sudo's access is currently unrestricted.
2. Users can remain with privileged permissions even after they leave the project.
3. There is no clear audit and control mechanism.
4. Permissions management is done manually.

**2. Why avoid sudo ALL?**

The ALL permission in sudo provides full access and raises several risks:

1. Users can become full root (sudo bash, sudo su).
2. They can read, modify, or delete sensitive files.
3. They can stop critical services: firewall, logging, auditing.
4. They can execute commands like other users, bypassing controls.
5. Auditing is almost impossible – everything is allowed and difficult to track.

**3. Objectives**

1. Granular control of sudo permissions using Unix groups.
2. Full automation with Ansible.
3. Easy to maintain and scalable solution.
4. Increased centralization and auditability.

**4. Proposed solution**

1. The servers will be organized by **roles** (e.g. kafka, tibco\_adapter, axway).
2. Each role has a **dedicated UNIX group**: kafka\_admins, tibco\_adapter\_admins, etc.
3. Users are defined centrally in a YAML file.
4. Ansible:
   1. Group servers by role;
   2. Create the necessary UNIX groups;
   3. Create users if they don't exist;
   4. Assign users to the appropriate groups;
   5. Apply suduers files;
   6. Remove users who no longer appear in the configuration.

5. Example of implementation

a) Declaration of hosts:

All:

children:

tibco\_adapter:

Hosts:

adapt01.domain.local

adapt02.domain.local

b) Declaring users in YAML:

tibco\_adapter\_admins:

-John

-Doe

-Max

c) Fisier sudoers:

/etc/sudoers.d/sudoers-tibco\_adapter

%tibco\_adapter\_admins ALL=(ALL) NOPASSWD: /usr/bin/systemctl \* tibco.service

**d) Ansible:**

1. Create the tibco\_adapter\_admins group if it doesn't exist;
2. Create and assign users to that group;
3. Create or copy the sudoers file to /etc/sudoers.d/.

**6. Proposed test;**

1. A test VM with access to Ansible is created.
2. Configure the necessary files on the Ansible server.
3. Apply the solution and check:
   1. User creation;
   2. UNIX group assignment;
   3. Set sudo permissions;
   4. Remove users who are no longer in YAML.

**7. Next steps**

1. Collaboration with other teams to identify current users.
2. Collection of command that require sudo.
3. Implementing the new sudoers file
4. Testing in test/dev environments.
5. Evaluation of results and adjustments.
6. Creation of final documentation.

**8. Benefits**

1. Centralized and automated permission control.
2. increased security.
3. Simplified and efficient audit.
4. Scalable solution for multiple teams and applications.
5. Easy to integrate with Git and Active Directory.