RED HAT®
CONSULTING

redhat.

Red Hat

Ansible Tower

RBAC Guide & Recommendations

Prepared by:

MATT NOLAN
PRINCIPAL ARCHITECT
STP - Cloud Practice
mnola

Role-Based Access Controls

Role-Based Access Controls (RBAC) are built into Tower and allow Tower administrators to delegate access to server inventories, organizations, and more. Administrators can also centralize the management of various credentials, allowing end users to leverage a needed secret without ever exposing that secret to the end user.

RBAC are easiest to think of in terms of Roles, which define precisely who or what can be seen, changed, or deleted. Starting with Tower 3.0, RBAC is best thought of as granting roles to users or teams for specific capabilities.

There are a few main concepts to become familiar with regarding Tower's RBAC design of roles, resources, and users.

1. Users can be members of a role, which gives them certain access to any resources associated with that role, or any resources associated with "child" roles.
2. A role is a collection of capabilities. Users are granted access to these capabilities and Tower's resources through the roles to which they are assigned to or through roles inherited through the role hierarchy.
3. Roles associate a group of capabilities with a group of users. All capabilities are derived from membership within a role.
4. Users receive capabilities only through the roles to which they are assigned or through roles they inherit through the role hierarchy.
5. All members of a role have all capabilities granted to that role. Within an organization, roles are relatively stable, while users and capabilities are both numerous and may change rapidly. Users can have many roles.

Applying RBAC

The following sections cover a recommended strategy for how to apply Tower's RBAC system in your environment.

Defining the System Administrator(s) (*aka...Superuser*)

A Tower system administrator should specify a user group (recommended to be a group mapped in from an external LDAP source) as being the *System Administrator* (also referred to as the Superuser).

- System administrators implicitly inherit all capabilities for all objects (read/write/execute) within the Tower environment.

Defining Organizations

When defining an organization, system administrators may specify the following roles:

- One or more users as organization administrators
- One or more users as organization members
- One or more teams as organization members

Users who are organization administrators will implicitly inherit all capabilities for all objects within that Tower organization.

Defining Projects in an Organization

When defining project(s) in an organization for which they are the administrator, system administrators, or organization administrators should specify:

- One or more teams that are project administrators

- One or more teams that are project members
- And one or more teams that may update the project from SCM, from among the teams that are members of that organization.

An individual user can be specified for the roles above, but it is recommended to leverage an LDAP based group/team mapping for ease of administration.

Users who are members of a project can view their project administrators.

Project administrators implicitly inherit the capability to update the project from SCM.

Administrators can also specify one or more users/teams (from those that are members of that project) that can use that project in a job template.

Creating Inventories and Credentials within an Organization

All access that is granted to use, read, or write credentials is handled through roles.

System administrators and organization administrators create inventories and credentials within organizations using their administrative capabilities.

When creating or modifying an inventory or a credential, System administrators and organization administrators specify one or more users/teams (from those that are members of that organization) to be granted the usage capability for that inventory or credential.

System administrators and organization administrators should specify one or more users/teams (from those that are members of that organization) that have the capability to update (dynamic or manually) an inventory. Administrators can also execute ad hoc commands for an inventory.

Defining Job Templates

System administrators, organization administrators, and project administrators, within a project under their administrative capabilities, create and modify new job templates for that project.

When creating or modifying a job template, administrators (Tower, organization, and project) can select among the inventory and credentials in the organization for which they have usage capabilities or they may leave those fields blank so that they will be selected at runtime.

Additionally, they may specify one or more users/teams (from those that are members of that project) that have execution capabilities for that job template. The execution capability is valid regardless of any explicit capabilities the user/team may have been granted against the inventory or credential specified in the job template. It is also important to note that Job templates inherit permissions from projects.

Applications

User View

A user can:

- See any organization or project for which they are a member
- Create personal credential objects which only belong to them
- See and execute any job template for which they have been granted execution capabilities

If a job template user has been granted execution capabilities and does not specify an inventory or credential, the user will be prompted at run-time to select among the inventory and credentials in the organization they own or have been granted usage capabilities.

Users that are job template administrators can make changes to job templates; however, to change the inventory, project, playbook, or credentials used in the job template, the user must also have the “Use” role for the project and inventory currently being used or being set.

Roles

As stated earlier in this documentation, all access that is granted to use, read, or write credentials is now handled through roles, and roles are defined for a resource.

A Singleton Role is a special role that grants system-wide permissions. Ansible Tower currently provides two built-in Singleton Roles but the ability to create or customize a Singleton Role is not supported at this time.

Common Team Roles - “Personas”

In order to allow Tower support personnel to focus more on ensuring Tower is available and well managed, it is recommended that Tower admins assign “Organization Owner/Admin” to specific users to create a new Organization and assign members from their team the required access as needed.

This better enables supporting individuals to focus more on maintaining uptime of the service and assisting users who are using Ansible Tower.

Below are common roles managed by the Tower organization:

System Role (for Organizations)	Common User Roles	Description
Owner	**Team Lead - Technical Lead**	This user has the ability to control access for other users in their organization. They can add/remove and grant users specific access to projects, inventories, and job templates.

		This user also has the ability to create/remove/modify any aspect of an organization's projects, templates, inventories, teams, and credentials.
Auditor	**Security Engineer - Project Manager**	This account can view all aspects of the organization in read-only mode. This may be good for a user who checks in and maintains compliance. This might also be a good role for a service account who manages or ships job data from Ansible Tower to some other data collector.
Member - Team	**All other users**	These users by default as an organization member do not receive any access to any aspect of the organization. In order to grant them access the respective organization owner needs to add them to their respective team and grant them Admin, Execute, Use, Update, Ad-hoc permissions to each component of the organization's projects, inventories, and job templates.
Member - Team "Owner"	**Power users - Lead Developer**	Organization Owners can provide "admin" through the team interface, over any component of their organization including projects, inventories, and job templates. These users are able to modify and utilize the respective component given access.
Member - Team "Execute"	**Developers - Engineers**	This will be the most common and allows the organization member the ability to execute job templates and read permission to the specific components. This permission applies to templates.

Member - **Team “Use”**	**Developers -** **Engineers**	This permission applies to an organization’s credentials, inventories, and projects. This permission allows the ability for a user to the respective component within their job template.
Member - **Team “Update”**	**Developers -** **Engineers**	This permission applies to projects. Allows the user to be able to run an SCM update on a project.

Necessary permissions to edit job templates

Users can edit fields not impacting job runs (non-sensitive fields) with a Job Template admin role alone. However, to edit fields that impact job runs in a job template, a user needs the following:

- **admin** role to the job template
- **use** role to related project
- **use** role to related inventory

In order to delegate *full* job template control (within an organization) to a user or team, you will need to grant the team or user all 3 organization-level roles:

- job template admin
- project admin
- inventory admin

This will ensure that the user (or all users who are members of the team with these roles) have full access to modify job templates in the organization.

For clarity of managing permissions, it is best-practice to not mix projects / inventories from different organizations.

RBAC permissions

Each role should have a content object, for instance, the org admin role has a content object of the org. To delegate a role, you need admin permission to the content object, with some exceptions that would result in you being able to reset a user's password.

Parent - is the organization.

Allow - is what this new permission will explicitly allow.

Scope - is the parent resource that this new role will be created on.
(Example: Organization.project_create_role)

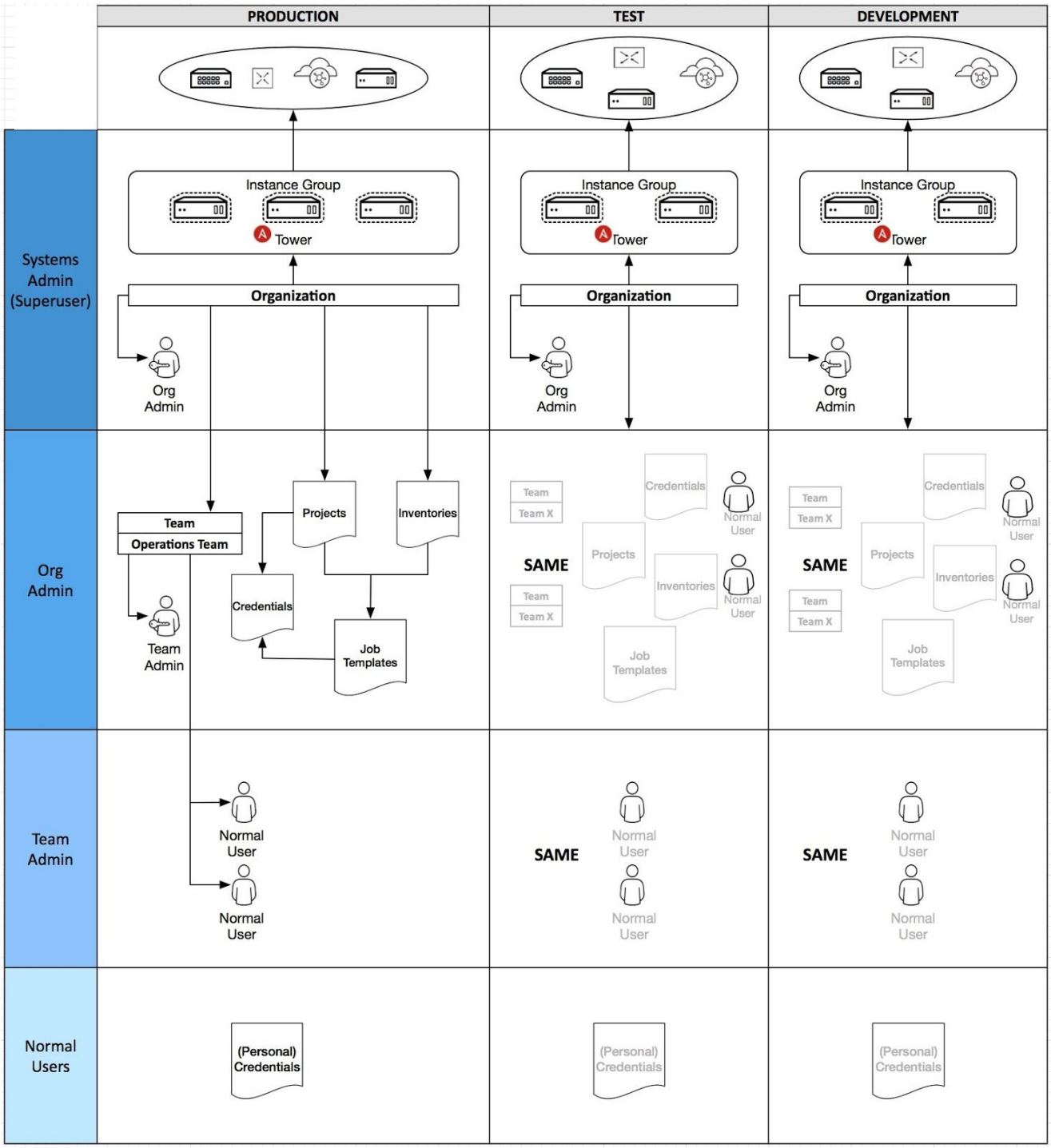
An assumption is being made that the creator of the resource should be given the admin role for that resource. If there are any instances where resource creation does not also imply resource administration, they will be explicitly called out.

Here are the rules associated with each admin type:

Admin Type	Rules
Project Admin	• **Allow:** Create, read, update, delete any project • **Scope:** Organization • **User Interface:** *Project Add Screen - Organizations*
Inventory Admin	• **Parent:** Org admin • **Allow:** Create, read, update, delete any inventory • **Scope:** Organization • **User Interface:** *Inventory Add Screen - Organizations*
Credential Admin	• **Parent:** Org admin • **Allow:** Create, read, update, delete shared credentials

	• **Scope:** Organization • **User Interface:** *Credential Add Screen - Organizations*
Notification Admin	• **Parent:** Org admin • **Allow:** Assignment of notifications • **Scope:** Organization
Workflow Admin	• **Parent:** Org admin • **Allow:** Create a workflow • Scope: Organization
Org Execute	• **Parent:** Org admin • **Allow:** Executing JTs and WFJTs • Scope: Organization

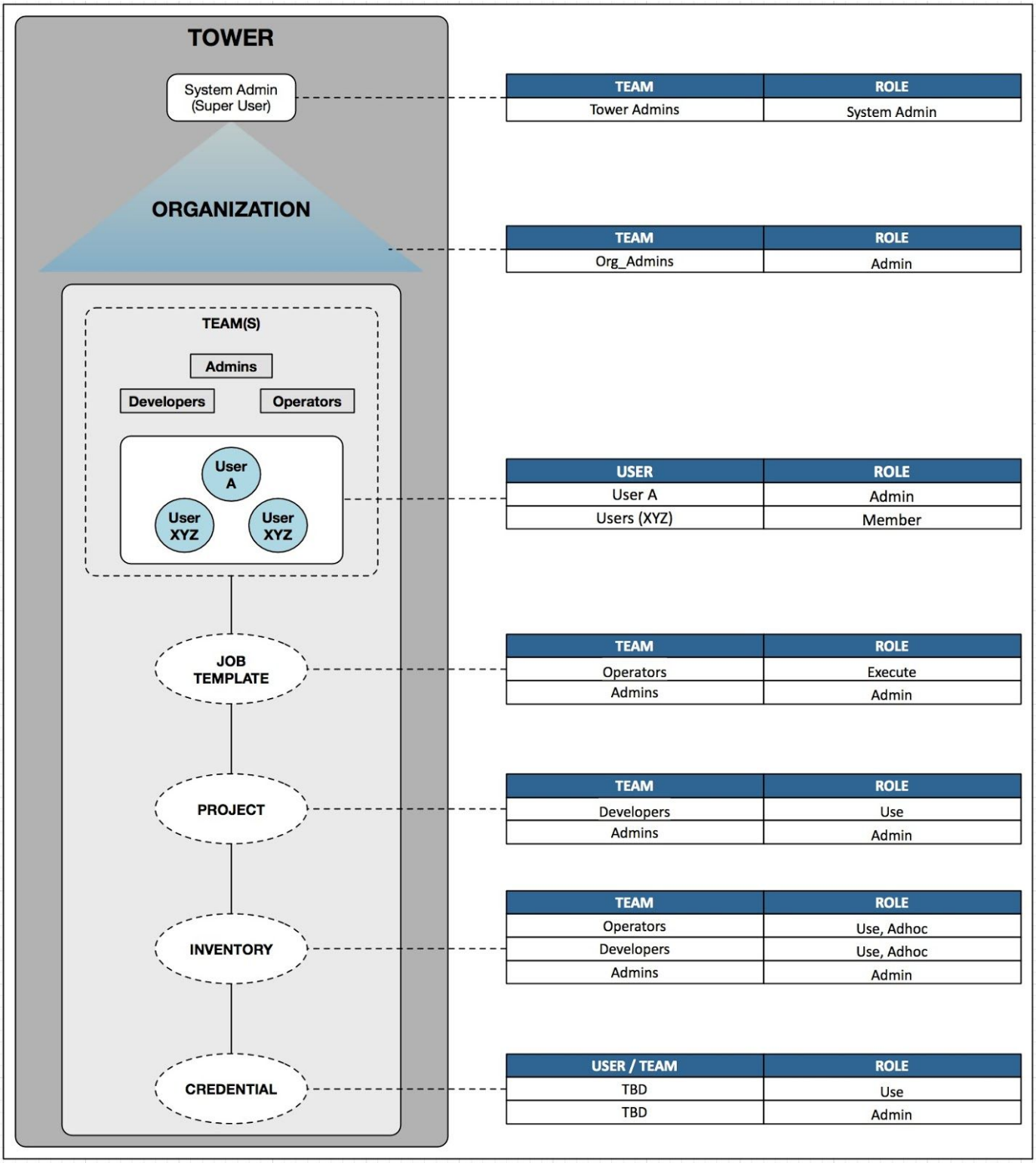
The following is a sample scenario showing an organization with its roles and which resource(s) each have access to:



Recommendations

- Pre-create the team and assign the roles ahead of time prior to mapping the LDAP group(s)
- Grant access at the team level and then grant admin access to a designated admin user if needed at the user level
- Users should create their own personal login credential to run job templates with and then when needed, be granted access to elevated credentials at the team level
- If possible, Red Hat recommends deploying separate Tower instances per environment. (Ex. 1 x Dev, 1 x Test, 1 x QA and 1 x Prod)

The following image illustrates the recommended roles to assign in a Tower environment:


TOWER
System Admin (Super User)
ORGANIZATION
TEAM(S)
Admins
Developers
Operators
User A
User XYZ
User XYZ
JOB TEMPLATE
PROJECT
INVENTORY
CREDENTIAL
TEAM
ROLE
Tower Admins
System Admin
TEAM
ROLE
Org_Admins
Admin
USER
ROLE
User A
Admin
Users (XYZ)
Member
TEAM
ROLE
Operators
Execute
Admins
Admin
TEAM
ROLE
Developers
Use
Admins
Admin
TEAM
ROLE
Operators
Use, Adhoc
Developers
Use, Adhoc
Admins
Admin
USER / TEAM
ROLE
TBD
Use
TBD
Admin

ROLES

SYSTEM ADMIN
Grants all capabilities for all objects in the Tower environment
Specifies user(s) as organization administrators
Specifies user(s) as organization auditors
Specifies user(s) or teams as organization members

ORGANIZATION LEVEL ROLES	
Admin	Grants all capabilities over all objects in that Tower org
Member	Allows users to view their org's administrator
Auditor	Grants read-only capability for all objects in that Tower org

USER ROLES	
Admin	- Grants all capabilities over all objects in that Tower org - Manages Team users and their associated team roles - Manages Team roles on resources for which the Team has been assigned the Admin role on
Member	- Allows users to inherit roles on team granted resources - Lets users see the Team users and associated Team roles

JOB TEMPLATE ROLES	
Admin	Grants full control over the job template
Use	Allows users to run the assigned job template

PROJECT ROLES	
Admin	Grants full control over the project
Use	Allows team members to use the project in a job template
Update	Grants update rights for the project from the SCM system

INVENTORY ROLES	
Admin	Grants modify rights to the inventory (CRUD operations)
Adhoc	Only allows single Adhoc commands against the inventory
Update	Allows a dynamic inventory to be updated from its external source
Use	Grants permission to run job templates against the inventory

CREDENTIAL ROLES	
Owner	Owns and manages all aspects of the credential
Use	Allows team members to use the credential in a project