Assignment Week 7

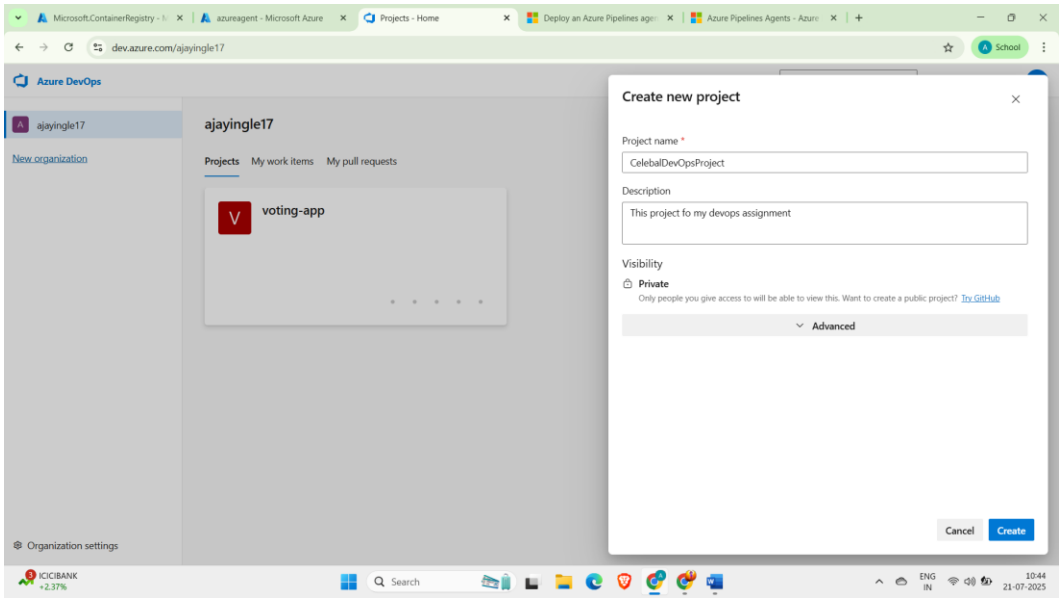
Task 1: Create a Project with Different User Groups and Implement Group Policies

Objective:
To manage access, roles, and responsibilities efficiently, Azure DevOps allows organizing users into groups with defined permissions. This ensures proper access control and governance for large teams.

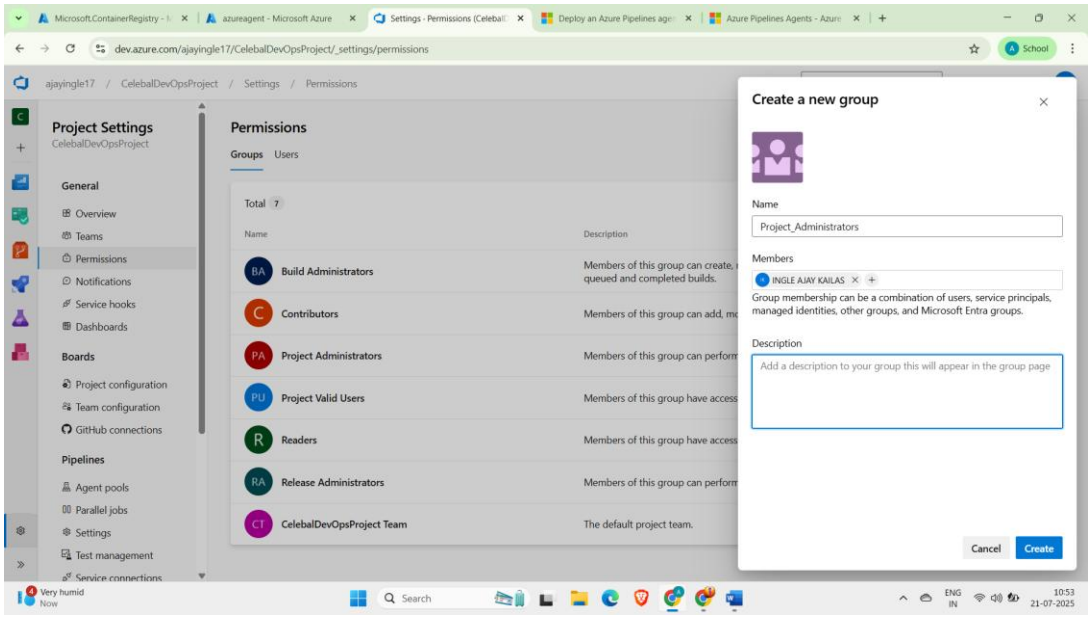
Theory:
User groups in Azure DevOps help you manage permissions in a scalable manner. Instead of assigning permissions individually, you define roles like Project Administrators, Contributors, and Readers. Permissions include code access, pipeline modification rights, and administrative settings.

Steps:

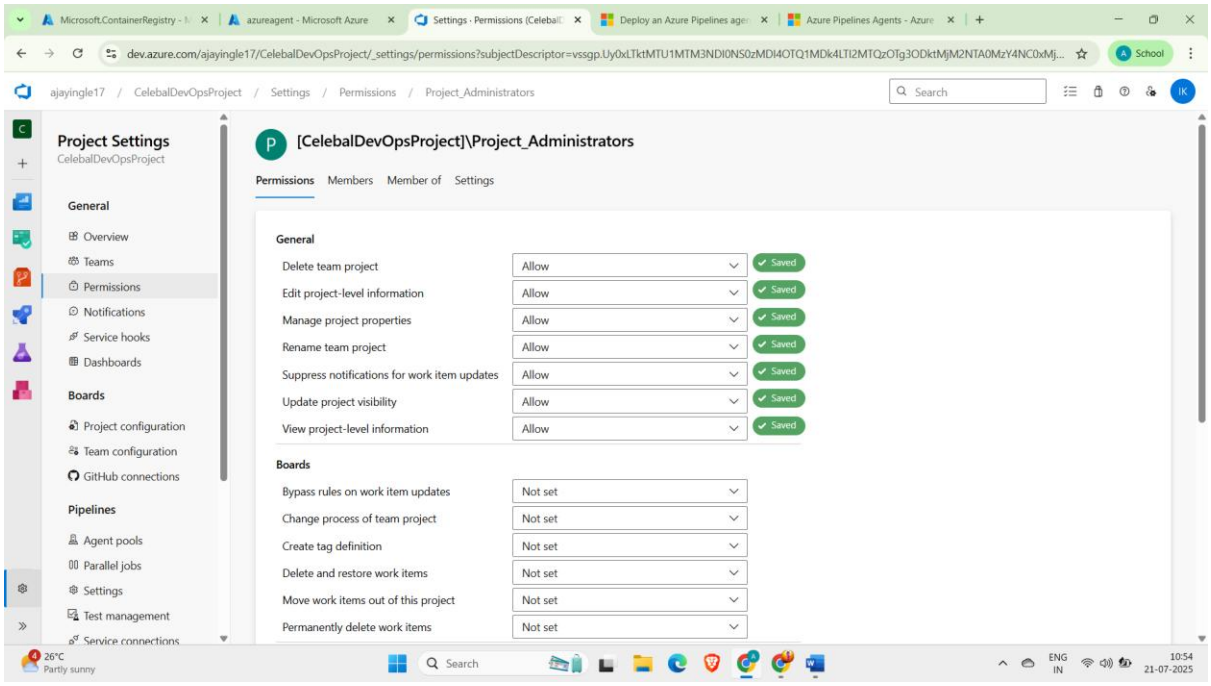
1. **Create a New Azure DevOps Project:**
    - **Visit https://dev.azure.com/**
    - **Click on New Project.**
    - **Enter a meaningful name like CelebalDevOpsProject.**
    - **Choose visibility: Private (recommended).**
    - **Click Create.**



2. **Create Custom Groups:**
    - **Navigate to Project Settings > Permissions.**
    - **Click New Group.**
    - **Create groups such as:**
        - **Project Administrators**
        - **Contributors**
        - **Viewers**
    - **Optionally, assign an image/icon for visual clarity.**
3. **Add Users to Groups:**
    - **Inside each group, click Members > Add.**
    - **Search and add users by email**



4. **Configure Group Permissions:**
    - **Click each group > Permissions tab.**
    - **For example:**
        - **Project Administrators: Full Access.**
        - **Contributors: Allow Contribute, Create Branch, Create Pull Requests, but Deny Force Push.**
        - **Viewers: Read access only.**



5. Best Practices:

- **Avoid direct permission assignment to users.**
- **Use inheritance and group membership to manage access.**

Task 2: Apply Branch Policies (Only Project Admins Access Master Branch)

Objective:
To enforce quality and security, branch policies restrict who can push to critical branches like master.

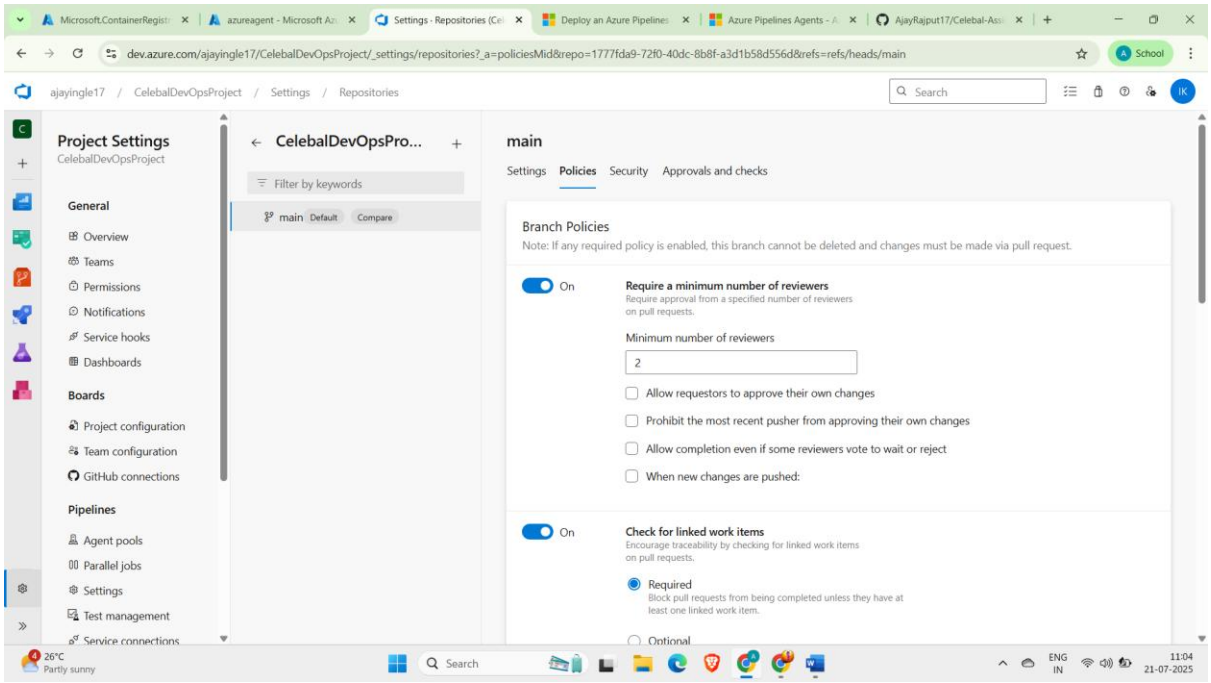
Theory:
Branch policies ensure that important branches are protected. They can enforce:

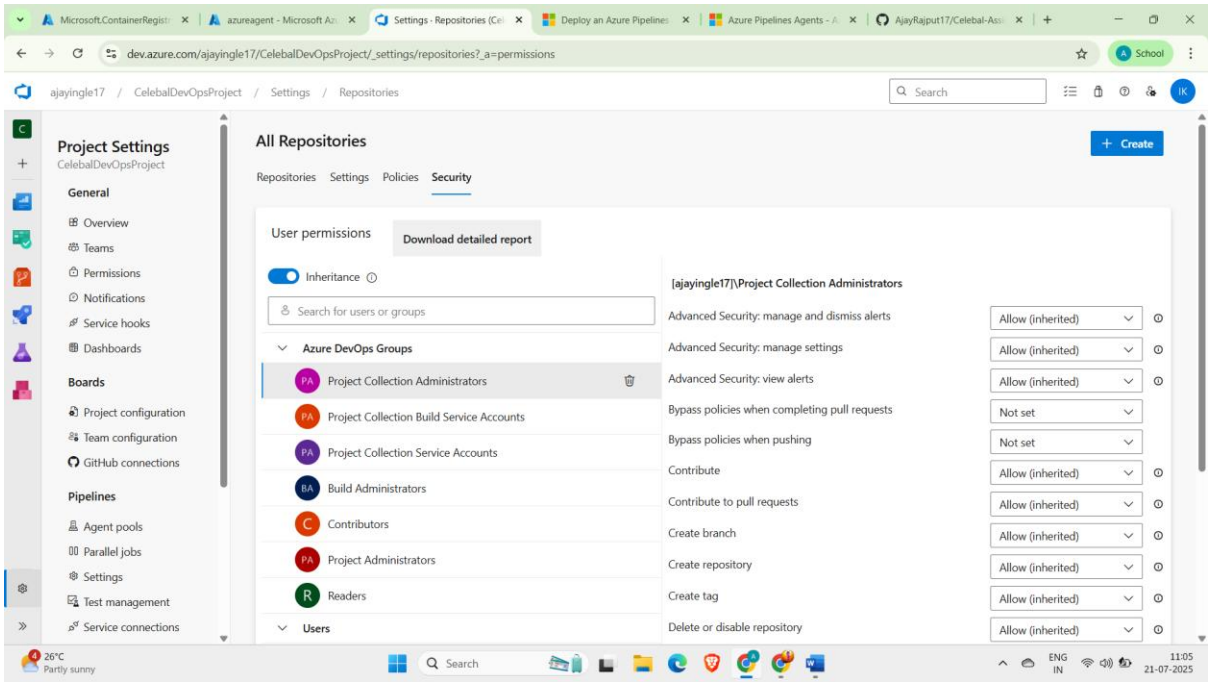
- **Review requirements before merging code**
- **Build validation**
- **Work item linking**
- **Restricting force-pushes or deletions**

Steps:

1. **Navigate to Repos > Branches.**
2. **Locate the master branch > click 3 dots > Branch Policies.**
3. **Enable the following:**
    - **Minimum number of reviewers: Set to 2.**
    - **Check for linked work items: Enabled.**
    - **Check for comment resolution: Enabled.**
    - **Limit merge types: Select Squash or Rebase.**
4. **Go to Project Settings > Repositories > master > Security:**
    - **Contributors: Deny Contribute, Allow Create PR.**
    - **Project Admins: Allow Contribute, Force push, Manage permissions.**

Result: Only project admins can push to master; contributors can only propose PRs.


dev.azure.com/ajayingle17/CelebalDevOpsProject/_settings/repositories?_a=policiesMid&repo=1777fda9-72f0-40dc-8b8f-a3d1b58d556d&refs=refs/heads/main
ajayingle17 / CelebalDevOpsProject / Settings / Repositories
Project Settings
CelebalDevOpsProject
General
Overview
Teams
Permissions
Notifications
Service hooks
Dashboards
Boards
Project configuration
Team configuration
GitHub connections
Pipelines
Agent pools
Parallel jobs
Settings
Test management
Service connections
CelebalDevOpsPro...
Filter by keywords
main Default Compare
main
Settings Policies Security Approvals and checks
Branch Policies
Note: If any required policy is enabled, this branch cannot be deleted and changes must be made via pull request.
On
Require a minimum number of reviewers
Require approval from a specified number of reviewers on pull requests.
Minimum number of reviewers
2
Allow requestors to approve their own changes
Prohibit the most recent pusher from approving their own changes
Allow completion even if some reviewers vote to wait or reject
When new changes are pushed:
On
Check for linked work items
Encourage traceability by checking for linked work items on pull requests.
Required
Block pull requests from being completed unless they have at least one linked work item.
Optional


dev.azure.com/ajayingle17/CelebalDevOpsProject/_settings/repositories?_a=permissions
ajayingle17 / CelebalDevOpsProject / Settings / Repositories
Project Settings
CelebalDevOpsProject
General
Overview
Teams
Permissions
Notifications
Service hooks
Dashboards
Boards
Project configuration
Team configuration
GitHub connections
Pipelines
Agent pools
Parallel jobs
Settings
Test management
Service connections
All Repositories
+ Create
Repositories Settings Policies Security
User permissions
Download detailed report
Inheritance
Search for users or groups
Azure DevOps Groups
Project Collection Administrators
Project Collection Build Service Accounts
Project Collection Service Accounts
Build Administrators
Contributors
Project Administrators
Readers
Users
[ajayingle17]\Project Collection Administrators
Advanced Security: manage and dismiss alerts
Allow (inherited)
Advanced Security: manage settings
Allow (inherited)
Advanced Security: view alerts
Allow (inherited)
Bypass policies when completing pull requests
Not set
Bypass policies when pushing
Not set
Contribute
Allow (inherited)
Contribute to pull requests
Allow (inherited)
Create branch
Allow (inherited)
Create repository
Allow (inherited)
Create tag
Allow (inherited)
Delete or disable repository
Allow (inherited)

Task 3: Apply Branch Security and Locks

Objective:
To prevent accidental changes and enforce administrative control, branches can be locked or restricted using security settings.

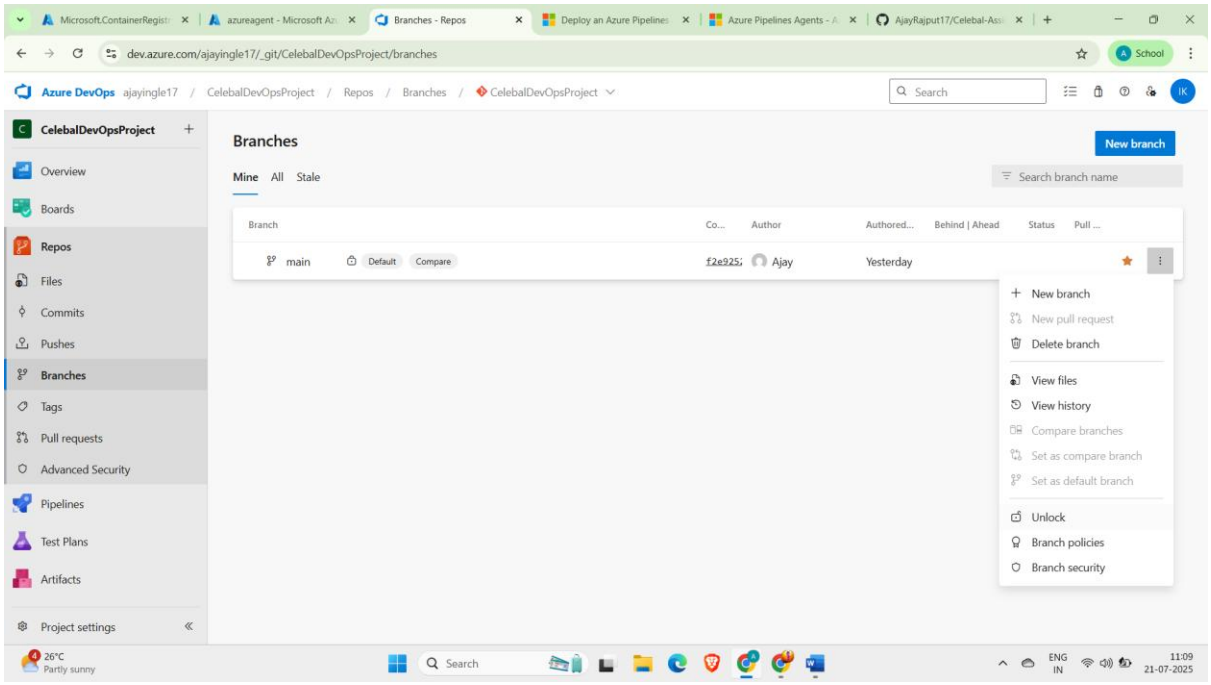
Theory:
Azure DevOps branch locking prevents any user from pushing to or deleting the branch until it is unlocked. Permissions like force-push and delete are managed via branch security settings.

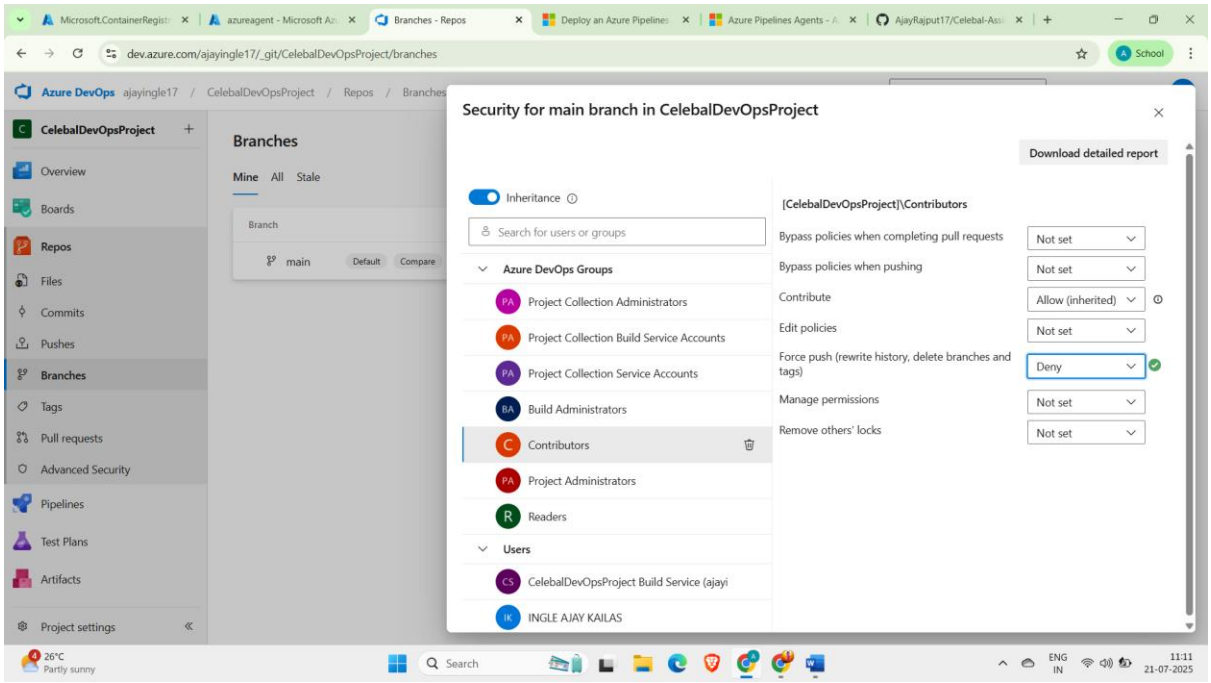
Steps:

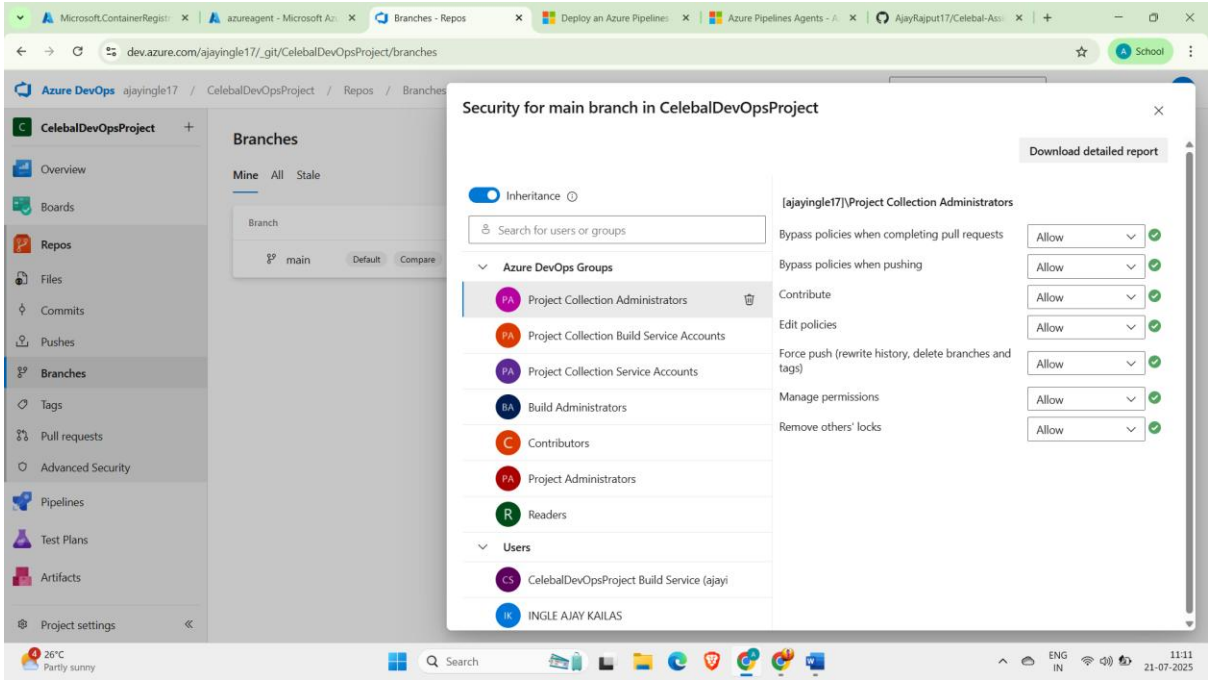
1. **Go to Repos > Branches > master.**
2. **Click More Options (•••) > Lock.**
    - **This prevents any push/delete operations.**
3. **To modify security:**
    - **Navigate to Project Settings > Repos > master > Security.**
    - **Adjust individual or group permissions as needed.**

Recommended Permissions:

- **Contributors: Deny Contribute, Deny Force Push**
- **Admins: Allow full access**




Security for main branch in CelebalDevOpsProject
Download detailed report
Inheritance
Search for users or groups
Azure DevOps Groups
Project Collection Administrators
Project Collection Build Service Accounts
Project Collection Service Accounts
Build Administrators
Contributors
Project Administrators
Readers
Users
CelebalDevOpsProject Build Service (ajayi
INGLE AJAY KAILAS
[CelebalDevOpsProject]\Contributors
Bypass policies when completing pull requests
Not set
Bypass policies when pushing
Not set
Contribute
Allow (inherited)
Edit policies
Not set
Force push (rewrite history, delete branches and tags)
Deny
Manage permissions
Not set
Remove others' locks
Not set


Security for main branch in CelebalDevOpsProject
Download detailed report
Inheritance
Search for users or groups
Azure DevOps Groups
Project Collection Administrators
Project Collection Build Service Accounts
Project Collection Service Accounts
Build Administrators
Contributors
Project Administrators
Readers
Users
CelebalDevOpsProject Build Service (ajayi
INGLE AJAY KAILAS
[ajayingle17]\Project Collection Administrators
Bypass policies when completing pull requests
Allow
Bypass policies when pushing
Allow
Contribute
Allow
Edit policies
Allow
Force push (rewrite history, delete branches and tags)
Allow
Manage permissions
Allow
Remove others' locks
Allow

Task 4: Apply Branch Filters and Path Filters

Objective: To control when and how builds are triggered based on branch or file-level changes.

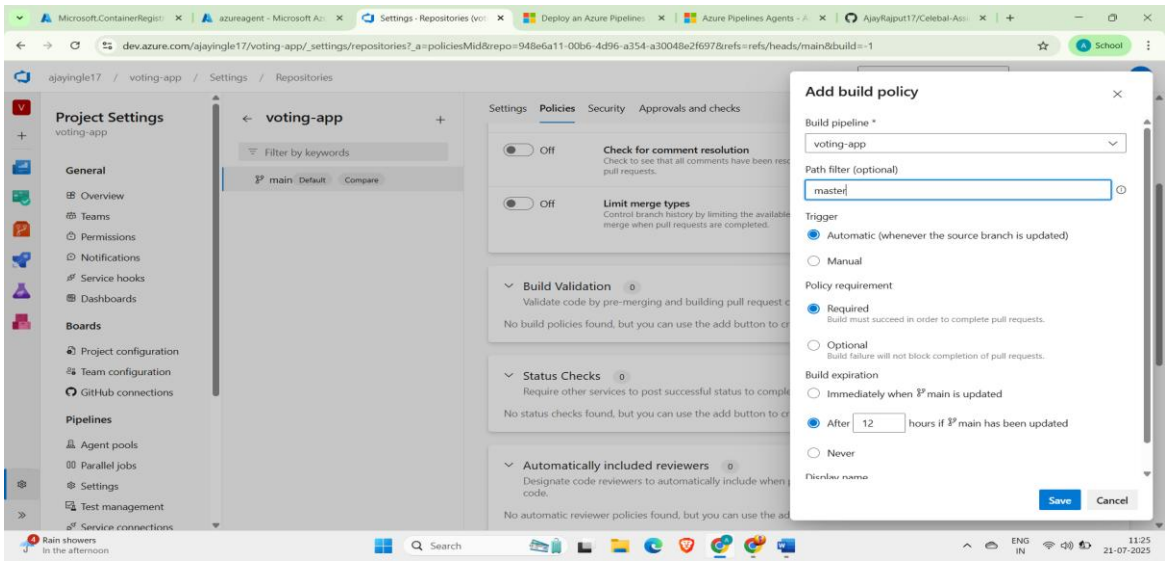
Theory: Filters improve CI/CD efficiency by ensuring builds only run on specific branches or directories (paths). Example: Only run tests when /src/ changes.

Steps:

1. **Go to Repos > Branches > master > Branch Policies.**
2. **Under Build Validation > Add Build Policy.**
3. **Set:**
    - **Branch Filters: master, develop, or feature/***
    - **Path Filters: Include /src/*, Exclude /docs/***

YAML Configuration Example:

```yaml
trigger:
  branches:
    include:
      - master
      - develop
  paths:
    include:
      - src/
    exclude:
      - docs/
```



Task 5: Apply a Pull Request

Objective: To collaborate securely, contributors use pull requests (PRs) to propose code changes instead of direct commits.

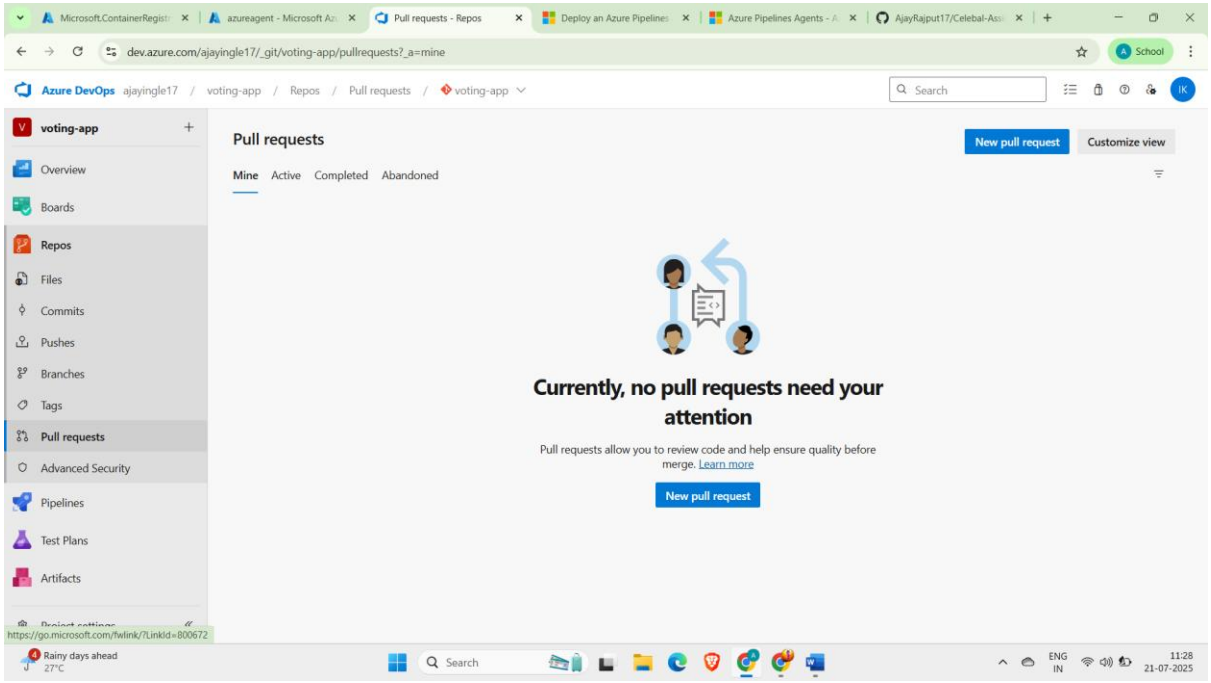
Theory: PRs allow peer review, automated validation, and safe integration. Combined with branch policies, this ensures quality and compliance.

Steps:

1. **Make changes in a local branch:**

```
git checkout -b feature/login
# make changes
git commit -m "#101 Login feature"
git push origin feature/login
```

2. ***Go to Repos > Pull Requests > New Pull Request.***
3. **Select source = feature/login, target = master.**
4. **Add reviewers > Click Create.**



Task 6: Apply Branch Policy and Security

Objective:
To safeguard the main branch (master) by applying a combination of branch policies and branch-level security settings, ensuring only authorized users (e.g., Project Administrators) can push or merge code directly, while contributors can only raise pull requests (PRs).

Theory:
Branch policies and branch security are two complementary features in Azure DevOps used to enforce code quality and protect critical branches:

- **Branch policies provide automatic checks and gatekeeping like requiring code reviews, linking work items, and successful builds before merges.**
- **Branch-level security controls who can perform actions such as pushing changes, deleting branches, or forcing updates.**

Together, they prevent unauthorized changes to sensitive branches like master and help maintain a clean, reviewed, and tested codebase.

Use Case:
This is especially important in production environments, where direct pushes could cause unreviewed or untested code to be deployed.

Steps to Configure:

1. Apply Branch Policy to master

1. **Navigate to Repos > Branches.**
2. **Click the 3-dot menu (•••) beside the master branch.**
3. **Select Branch Policies.**

Configure the following policies:

- **Minimum number of reviewers: Set to 2.**
- **Check for linked work items: Enforce linking work items with each PR.**
- **Check for comment resolution: Ensure all review comments are resolved before merge.**
- **Build Validation: Add a build pipeline to ensure the code compiles and tests pass.**
- **Merge types allowed: Only allow Squash merges to maintain clean history.**

Click Save Changes.

---

2. Configure Branch Security

1. **Go to Project Settings > Repositories > master > Security.**
2. **Select the Contributors group.**
3. **Set permissions:**
    - **Contribute → Deny**
    - **Force push (rewrite history, delete branches) → Deny**
    - **Create branch → Allow**
    - **Create pull requests → Allow**
4. **Select the Project Administrators group:**
    - **Set all relevant permissions to Allow, including:**
        - **Contribute**
        - **Bypass policies when pushing**
        - **Force push**
        - **Manage permissions**

This ensures only administrators can directly commit, while contributors must go through the PR and review pipeline.

Result:

✅ **Contributors can:**

- **Create new branches**
- **Push to their own branches**
- **Create pull requests targeting master**

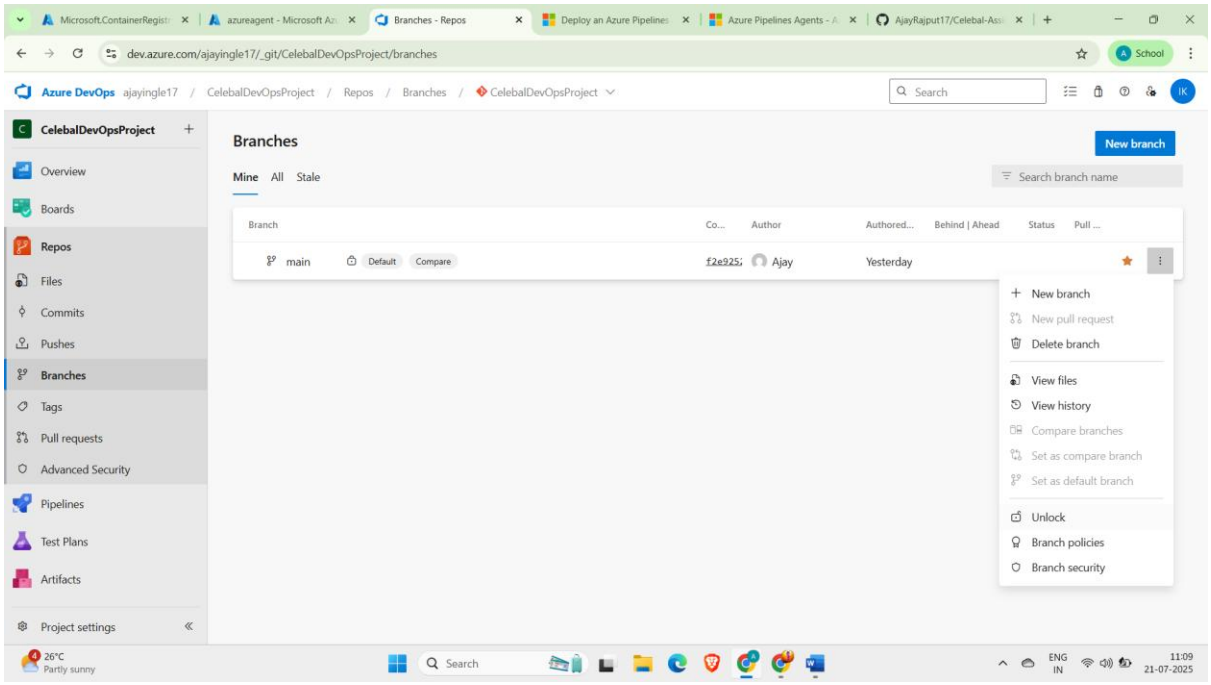
🚫 **Contributors cannot:**

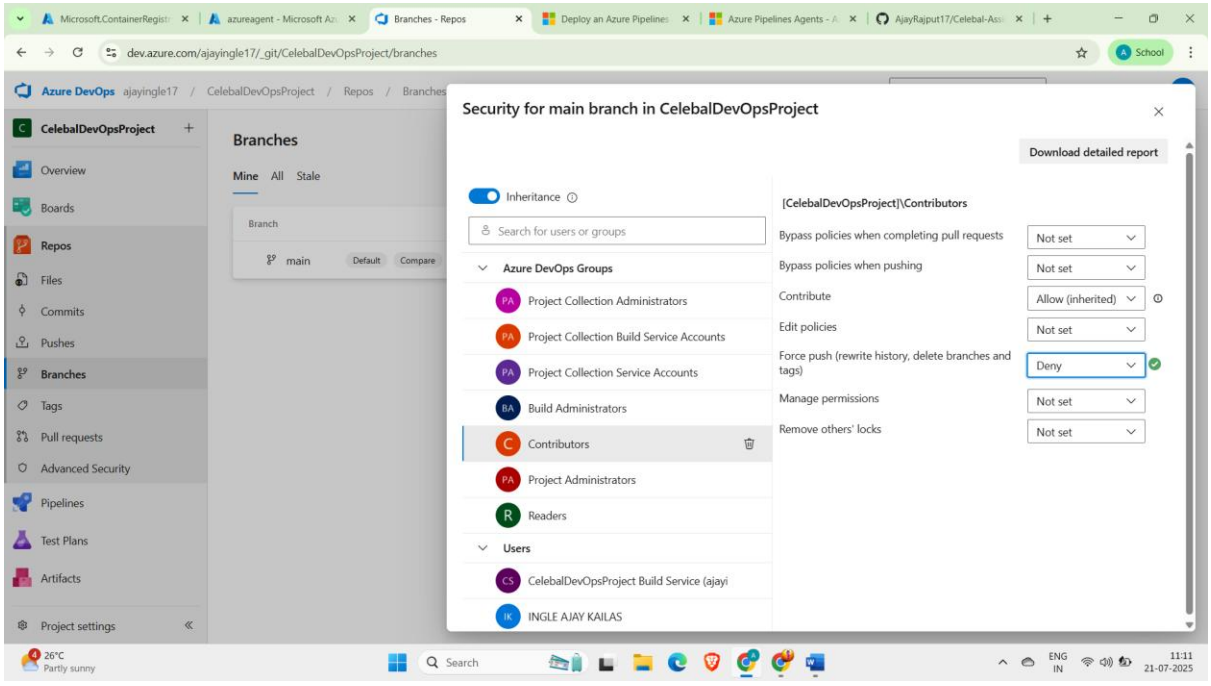
- **Push directly to master**
- **Merge without completing the required policies (review, build, etc.)**

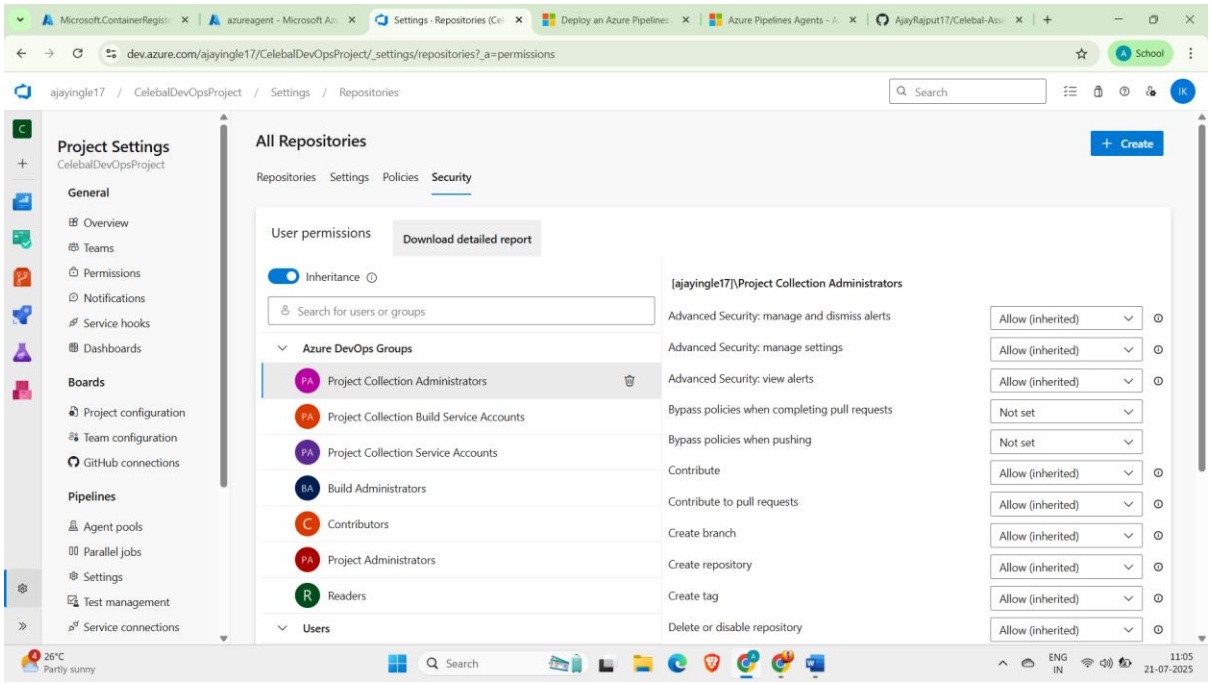
Best Practices:

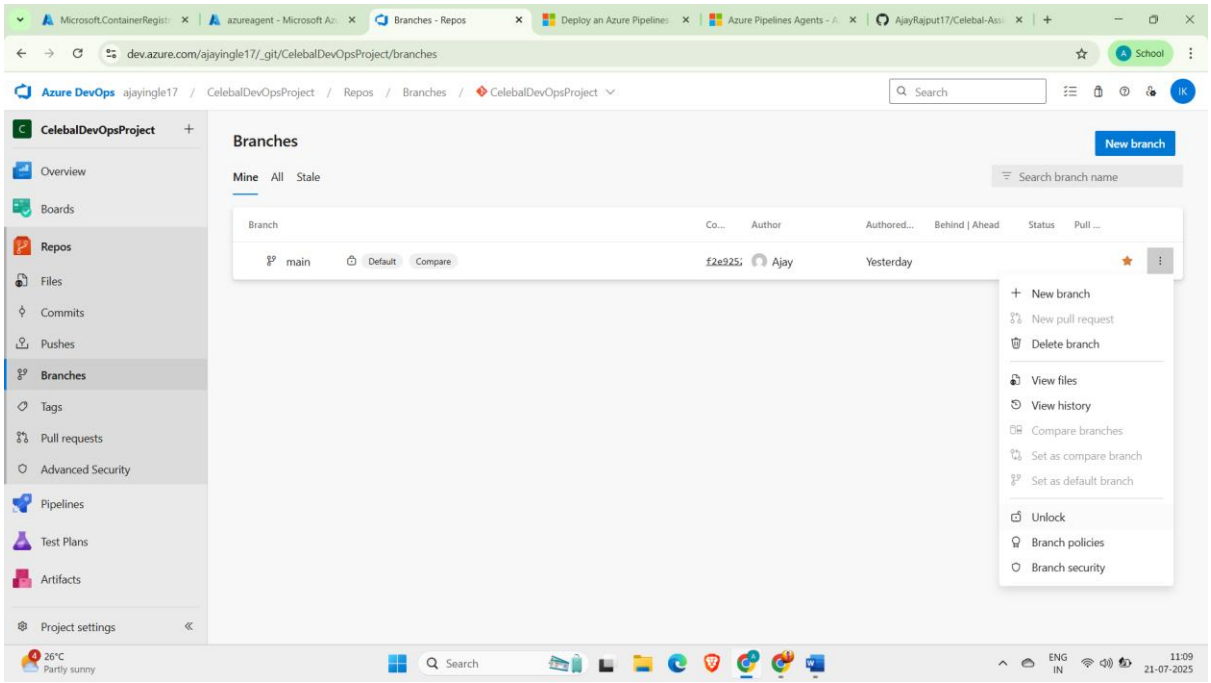
- **Never give direct write access to master unless absolutely necessary.**

- **Regularly review branch permissions as team roles evolve.**
- **Audit pull requests to monitor for policy bypassing (if any).**






dev.azure.com/ajayingle17/CelebalDevOpsProject/_settings/repositories?_a=permissions
ajayingle17 / CelebalDevOpsProject / Settings / Repositories
Project Settings
CelebalDevOpsProject
General
Overview
Teams
Permissions
Notifications
Service hooks
Dashboards
Boards
Project configuration
Team configuration
GitHub connections
Pipelines
Agent pools
Parallel jobs
Settings
Test management
Service connections
All Repositories
Create
Repositories
Settings
Policies
Security
User permissions
Download detailed report
Inheritance
Search for users or groups
Azure DevOps Groups
Project Collection Administrators
Project Collection Build Service Accounts
Project Collection Service Accounts
Build Administrators
Contributors
Project Administrators
Readers
Users
[ajayingle17]\Project Collection Administrators
Advanced Security: manage and dismiss alerts
Allow (inherited)
Advanced Security: manage settings
Allow (inherited)
Advanced Security: view alerts
Allow (inherited)
Bypass policies when completing pull requests
Not set
Bypass policies when pushing
Not set
Contribute
Allow (inherited)
Contribute to pull requests
Allow (inherited)
Create branch
Allow (inherited)
Create repository
Allow (inherited)
Create tag
Allow (inherited)
Delete or disable repository
Allow (inherited)


dev.azure.com/ajayingle17/_git/CelebalDevOpsProject/branches
Azure DevOps ajayingle17 / CelebalDevOpsProject / Repos / Branches / CelebalDevOpsProject
CelebalDevOpsProject
Overview
Boards
Repos
Files
Commits
Pushes
Branches
Tags
Pull requests
Advanced Security
Pipelines
Test Plans
Artifacts
Project settings
Branches
New branch
Mine
All
Stale
Search branch name
Branch
Co...
Author
Authored...
Behind | Ahead
Status
Pull ...
main
Default
Compare
f2e925;
Ajay
Yesterday
New branch
New pull request
Delete branch
View files
View history
Compare branches
Set as compare branch
Set as default branch
Unlock
Branch policies
Branch security

Task 7: Apply Triggers in Build and Release Pipelines

Objective: Automatically trigger pipelines when changes are pushed to specified branches.

Theory: Triggers ensure automated testing/building occurs without manual intervention. They improve CI/CD flow and catch errors early.

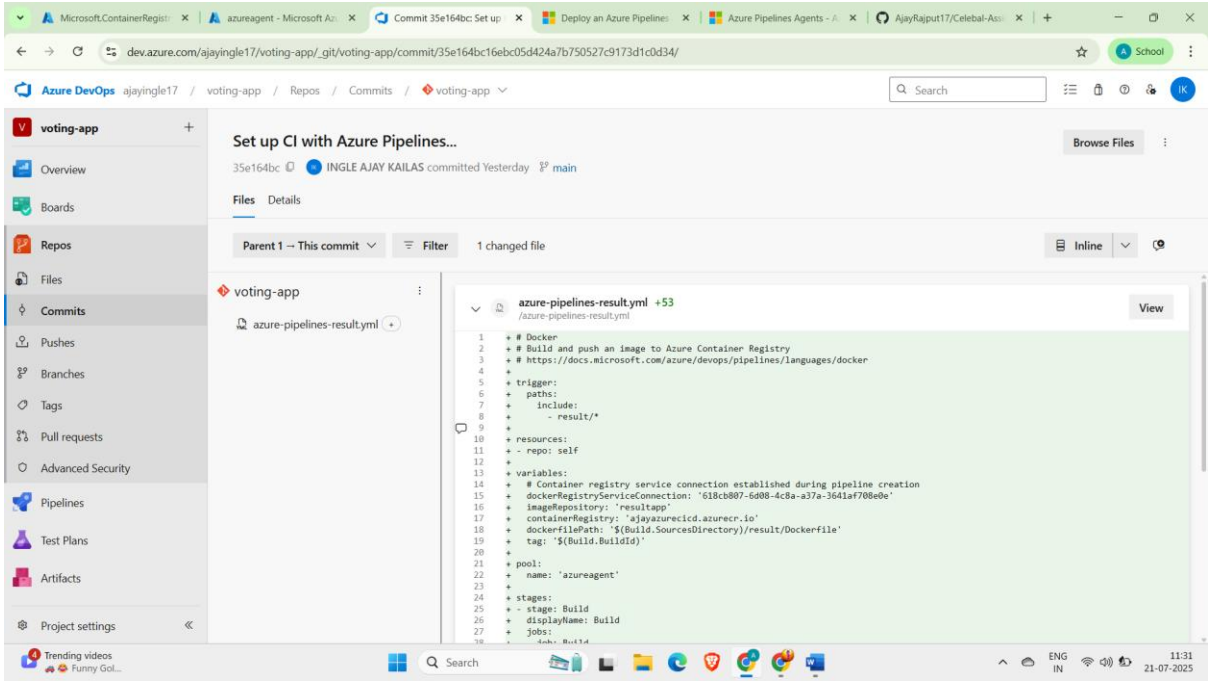
Steps:

1. **Create azure-pipelines.yml in repo root:**

```yaml
trigger:
  branches:
    include:
      - master
      - feature/*
```

2. **For release pipelines:**

- **Navigate to Pipelines > Releases > Edit.**
- **Enable Continuous Deployment Trigger.**



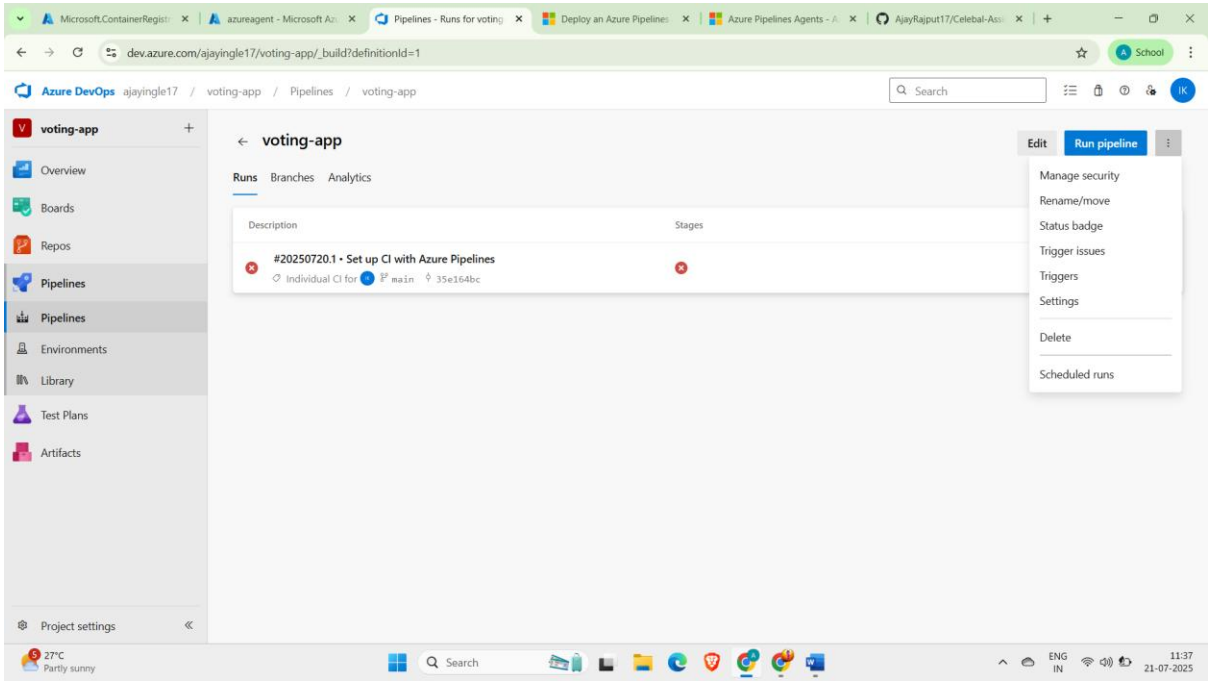
Task 8: Apply Gates to the Pipeline

Objective: Use automated gates (checks) before deploying to production.

Theory: Gates add quality checks like querying a REST API, monitoring status, or checking work items. They are part of release pipeline pre-deployment conditions.

Steps:

1. **Go to Pipelines > Releases > Edit Stage.**
2. **Click Pre-deployment Conditions > Gates.**
3. **Configure one or more:**
    - **REST API status check**
    - **Query Azure Monitor logs**
    - **Check for unresolved work items**
4. **Set sampling interval and timeout.**



Task 9: Apply Branch Security (Contributors PR Only, No Direct Merge)

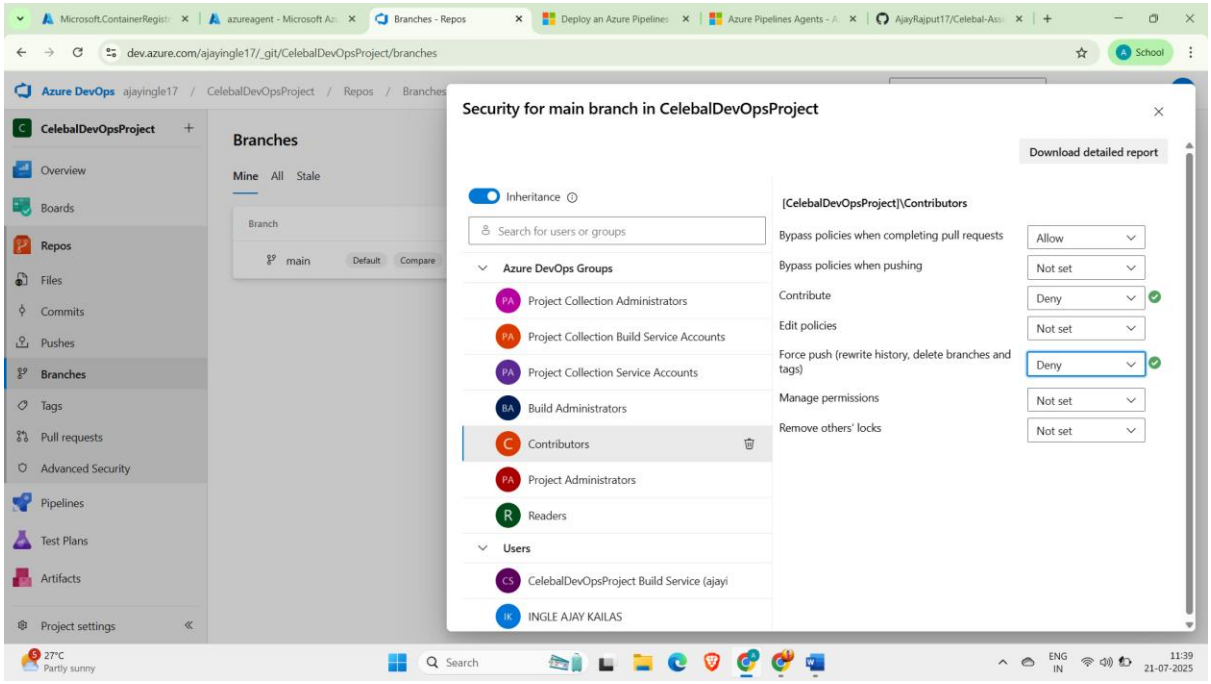
Objective: To enforce peer review and avoid accidental commits to master.

Theory: By restricting direct access, you ensure only validated code reaches production branches. PR-based workflows are more secure.

Steps:

1. **Project Settings > Repos > master > Security:**
    - **Contributors:**
        - **Deny: Contribute, Force Push**
        - **Allow: Create branch, Create Pull Requests**
    - **Admins:**
        - **Allow all**

Result: Contributors can create and submit PRs but cannot push to master directly.



Task 10: Apply Branch Policy and Security (Detailed)

📘 Theoretical Background:

Branch policies and security settings are crucial in maintaining code quality and protecting critical branches (like main or master) in collaborative development environments. In Azure DevOps, branch policies enforce rules such as mandatory pull request reviews, build validations, and restricted push access, which together reduce the risk of introducing bugs or unauthorized code changes into production.

Key Concepts:

- **Branch Policies: Control who can push, merge, and enforce reviews/builds.**
- **Security Permissions: Define what contributors, readers, and admins can do in a repo.**
- **Pull Requests (PRs): Used to review and approve changes before merging to the main branch.**
- **Protected Branch: A branch where direct pushes are blocked; changes must go through a PR process.**

🛠️ Objective:

Ensure that:

- **Contributors can initiate pull requests (PRs).**
- **Contributors cannot merge PRs directly to protected branches (like main).**
- **Contributors cannot push directly to the protected branches.**
- **Proper branch security and permission settings are applied.**

✅ Step-by-Step Implementation:

🔹 Step 1: Navigate to the Repository Settings

1. **Go to your Azure DevOps project.**
2. **Click on Repos in the left sidebar.**
3. **Select the repository you want to secure.**

4. Click the gear icon (⚙️) at the bottom left corner to open Repository Settings.

🔹 Step 2: Configure Branch Policies

1. Under Repositories → Branches, locate your main or master branch.
2. Hover over it and click on the three dots (⋮) → select Branch policies.
3. Apply the following settings:
    - ✅ Require a minimum number of reviewers → set to 1 or 2.
    - ✅ Check for linked work items (Optional but good for traceability).
    - ✅ Limit merge types → disable options like "Allow bypass."
    - ✅ Build validation (optional): Add a build pipeline to run on PRs.

🔹 Step 3: Set Branch Security

1. Still under Branches, click three dots (⋮) next to main → Branch security.
2. Under Users and Groups, select the Contributors group.
3. Apply the following permissions:
    - Contribute → Deny
    - Force push → Deny
    - Create branch → Allow
    - Read → Allow
    - Contribute to pull requests → Allow

🔐 Note: Denying Contribute prevents direct push to the branch; Contribute to pull requests allows PR submissions.

🔹 Step 4: Test the Configuration

1. Sign in with a test Contributor account or ask a teammate.
2. Try pushing directly to the main branch → should fail.
3. Create a new branch and submit a PR to main → PR should succeed, but merging should require approval from another role (like Project Admin).

📌 **Best Practices:**

- **Always enforce PR approvals and build validations on main/production branches.**
- **Limit merge permissions to senior developers or DevOps engineers.**
- **Regularly audit permissions to avoid accidental privilege escalations.**
- **Use branch naming conventions (e.g., feature/*, bugfix/*) and policies accordingly.**
