**The following will be covered in this Lab**

Exercise 1– Manage Repos

Task 1: Creating a new repo from Azure DevOps .

Task 2: Deleting and renaming Git repos

Exercise 2 : Working with Branches

Task 1: Creating a new Branch

Task 2: Deleting a branch

Task 3: Locking a Branch

Task 4: Branch Policies

Task 5: Set Branch policies reviewers

Exercise 3 Branch Policies – Pull request.

Task1: Creating new pull request

Task 2: Managing Pull Requests

Task3 : : Managing Git branch and pull request policies

Task 4: Branch Policies - Check for Comment resolution

**Exercise 1– Manage Repos**

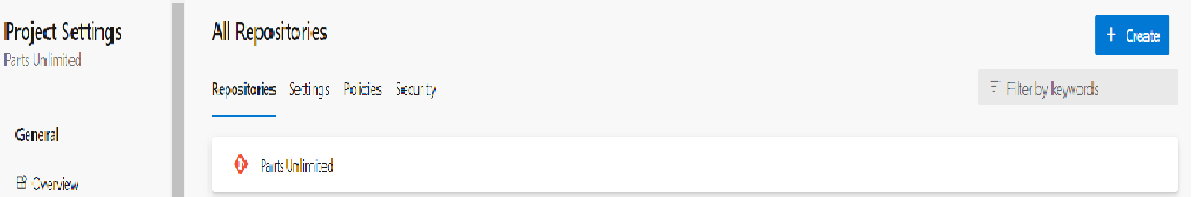
**Task 1: Creating a new repo from Azure DevOps .**

1. From the project Add dropdown, select **new** repository.

A screenshot of a computer

Description automatically generated

Or you can create repository from project settings.



1. Set the Repository with a name. Note that you also have the option to create a file named README.md. This would be the default markdown file that is rendered when someone navigates to the repo root in a browser. Additionally, you can preconfigure the repo with a .gitignore file. This file specifies which files, based on naming pattern and/or path, to ignore from source control. There are multiple templates available that include the common patterns and paths to ignore based on the project type you are creating. Click Create

A screenshot of a computer

Description automatically generated

1. That's it. Your repo is ready. Optionally You can now clone it with Visual Studio or your tools of choice.

**Task 2: Deleting and renaming Git repos**

1. Sometimes you'll have a need **to rename or delete** a repo, which is just as easy. **Open Project setting**

A screen shot of a computer

Description automatically generated

1. Select Repositories under Repo

A close-up of a text

Description automatically generated

1. From the New Repo context menu, select Delete repository. Alternatively, you could rename it here.

A screenshot of a computer

Description automatically generated

**Exercise 2 : Working with Branches**

You can manage the work in your Azure DevOps Git repo from the Branches view on the web.

You can also customize the view to track the branches you care most about so you can stay on top

of changes made by your team.

Committing changes to a branch will not affect other branches and you can share branches with others

without having to merge the changes into the main project. You can also create new branches to isolate changes

for a feature or a bug fix from your master branch and other work. Since the branches are lightweight,

switching between branches is quick and easy.

Git does not create multiple copies of your source whenworking with branches, but rather uses the

history information stored in commits to recreate thefiles on a branch when you start working on it.

Your Git workflow should create and use branches formanaging features and bugfixes. The rest of the

Git workflow, such as sharing code and reviewingcode with pull requests, all work through branches.

Isolating work in branches makesit very simple to change what you are working on by simply changing your current branch.

**Task 1: Creating a new Branch**

1. Switch to the Azure DevOps browser tab.
2. Navigate to Repos | Branches. Click New branch.

A blue rectangle with white text

Description automatically generated

1. Enter a name of "release" for the new branch. Optionally Use the Work items to link dropdown to select one or more work items to link to this new branch. Click Create branch to create it.

A screenshot of a computer screen

Description automatically generated

1. After the branch has been created, it will be available on the list.

A white rectangular object with a white background

Description automatically generated with medium confidence

**Task 2: Deleting a branch**

1. Return to Azure DevOps and click the Delete branch button from the more actions drop down to delete it.

A screenshot of a computer

Description automatically generated

1. You can Restore branch if you want by ,searching for an exact branch name. Select Restore branch as shown below.

A screenshot of a computer

Description automatically generated

**Task 3: Locking a Branch**

Locking is ideal for preventing new changes that might conflict with an important merge or to place a branch into a read-only state. Alternatively, you can use branch policies and pull requests instead of locking if you just want to ensure that changes in a branch are reviewed before they are merged. Locking does not prevent cloning of a repo or fetching updates made in the branch into your local repo. If you lock a branch, share with your team the reason why and make sure they know what to do to work with the branch after it is unlocked.

1. From the main branch context menu, select Lock

A screenshot of a computer

Description automatically generated

1. The branch is now locked

A screenshot of a computer

Description automatically generated

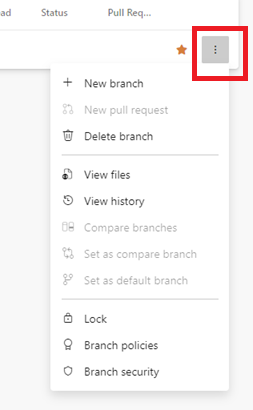
**Task 4: Branch Policies**

1. Select a branch

A screenshot of a computer

Description automatically generated

1. Click on context menu and select branch policies



1. Keep it in default settings

A screenshot of a computer screen

Description automatically generated

1. Navigate back to Repo/Main branch
2. Edit ReadMe file and Commit
3. Commits successfully

**Task 5: Set Branch policies reviewers**

1. Launch branch policies and enable reviewers

A screenshot of a computer

Description automatically generated

1. Navigate back to Repo/Main branch
2. Edit ReadMe file and Commit
3. Try to commit. Now observe the behavior

A screenshot of a computer

Description automatically generated

**Exercise 3 Branch Policies – Pull request.**

Pull requests let your team give feedback on changes in feature branches before merging the code into the master branch. Reviewers can step through the proposed changes, leave comments, and vote to approve or reject the code. Azure DevOps provides a rich experience for creating, reviewing, and approving pull requests.

**Task1: Creating new pull request**

1. Create a new branch out of Main and link workitems

A screenshot of a computer

Description automatically generated

1. We just created a new branch and make some change to readme . Now we need to commit that change to the new branch and push it to the server. Once there, we can create a pull request so that the branch can be merged with the main. A screenshot of a computer

   Description automatically generated
2. Edit the readme file in Feature 1 branch

A screenshot of a computer

Description automatically generated

1. Commit the changes by clicking commit

A screenshot of a computer

Description automatically generated

1. Confirm by clicking the commit button. Optionally link wok items

A screenshot of a computer

Description automatically generated

1. Click on the Create a Pull Request button in the information message to create a pull request

A screenshot of a computer

Description automatically generated

1. The New Pull Request form should already contain the key information reviewers will need, as well as who those reviewers should be (if any). If not, select release as the branch to merge into master.

You can customize any of this, and some of it may be required based on policy. Also note that the work item associated with the branch when created is referenced. Click Create.

A screenshot of a computer

Description automatically generated

**Task 2: Managing Pull Requests**

1. The Overview tab of Pull request contains all of the key information specified in the creation form, as well as options to approve and complete the request.

Select the Files tab to review the files involved in the commit.

Click the Add comment button next to the source file. Enter a comment using markdown and click Comment to save it. Note that there is a live preview of your comment before you commit to it. Not that any comment entered have to be resolved

Select the Commits tab, where you can review the commits made to the branch.

Since everything seems to be in order, Approve the pull request

A screenshot of a computer

Description automatically generated

1. Approve the pull request

A screenshot of a computer

Description automatically generated

1. Since the approvers have signed off, Complete the pull request.

A screenshot of a computer

Description automatically generated

1. You can accept the default messages in the pull request dialog. The first option is to complete the work items linked to the branch being merged. Note that you also have the option to delete the target branch after merging, as well as the ability to modify the commit message. From the drop down, you can also perform a squash changes during the merge. Squash merging is a merge option that allows you to condense the Git history of topic branches when you complete a pull request. Instead of each commit on the topic branch being added to the history of the default branch, a squash merge takes all the file changes and adds them to a single new commit on the default branch. Squash merging keeps your default branch histories clean and easy to follow without demanding any workflow changes on your team. Click Complete merge

A screenshot of a computer screen

Description automatically generated

1. When the pull requests complete. PR should be marked completed.

Click back on the Overview tab. You can review the pull request, as well as Cherry-pick or Revert it if needed. Cherry-picking is the process of selecting specific commits from one branch to apply to another, conceptually similar to a copy/paste operation.

Under Work Items, note that the work item has now been marked as Completed

A screenshot of a computer

Description automatically generated

1. Create pull request to merge the change from Feature 1 to Main and follow the same process till approval

A white rectangular box with black text

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a video

Description automatically generated

Click on Approve

And then Auto COmplete

A blue rectangle with white text and a lightning bolt

Description automatically generated

Click on set auto-complete

A screenshot of a computer

Description automatically generated

**Task3 : : Managing Git branch and pull request policies**

As projects and teams scale in complexity, it becomes help to automate more of the processes put in place to ensure quality.

1. Open Project Settings
2. Select Repositories under Repos
3. Select the Tailspin repo. Like everything else in Azure DevOps, you can manage security to a great level of detail. The **Settings tab** allows you to set some useful policies, such as whether or not you want to allow users to fork the repo, whether discussed work items are automatically linked, and so on
4. Select policies tab. Here are additional policies available at the repository level

A screenshot of a computer

Description automatically generated

1. Navigate back and select a repository and policies tab. Azure DevOps branch policies are very effective in enforcing a level of quality control in the repo. For example, you can control pull requests by requiring a minimum number of reviewers, checking for linked work items, requiring that all comments have been resolved, and more. You can even require validation through a successful build and configure external approval services. If there are certain sections of code that require their own approvers to be included, you can include them here as well.

A screenshot of a computer

Description automatically generated

1. Scroll down and select the main branch. Like the repo, you have a great deal of control over its security settings. You can also define a wide variety of policies for the branch to enforce. Check Require a minimum number of reviewers. By default, this requires at least two reviewers to approve a pull request, and also requires (by default) that the original author is not one of them.

A screenshot of a computer screen

Description automatically generated

1. Add yourself as a Reviewer and set the Path filter to "/Tailspin.SpaceGame.Web/Controllers"\*. Having the requirement of Required ensures that you will be required to sign off on any changes proposed to the controllers code in the web app. Click Save

A screenshot of a computer screen

Description automatically generated

Task 4: Branch Policies - Check for Comment resolution

1. Select Branch policies of Main Branch
2. Select Policies tab and enable comment resolution

A screenshot of a computer

Description automatically generated