**Boards:**

* Epic:
* Larger part of the project.
* Encapsulates user stories.
* Usually user stories are child of epic
* Steps:

Boards>New work items>Epic

* User stories:
  + Small task / feature / requirement.
  + Child of Epic
  + Steps:

Boards>New Work items>User stories

* Linking objects.

Select any epic/user story/work item, navigate to bottom right and select related items. Select how the other work item is related to the selected one and select the other work item. Save

* Boards:
  + It is used to plan and track the task/features.
  + You can create the user stories/features from here as well.
* Sprints:
  + Used to track the and assign the work.
* User Management:
  + Only Microsoft accounts can be added as a user to azure devops.
  + Org settings> Users> Create user
  + The user will receive invite to access this organization/project
  + You can assign access to some specific projects with required access levels (stakeholder, basic, basic + test) and required permissions (project administrators etc.)
  + You can also integrate the azure devops with azure ad. In that case, you don’t need Microsoft account and can directly assign the permissions to azure ad users.
* Dashboards:
  + Present under Overview section of the project
  + You can create your own dashboard using various widgets
  + Some popular widgers are as below.
    - Burndown : Time remaining
    - Lead time
    - Cycle time
    - Velocity: capacity to deliver work sprint after sprint
* Queries:
  + Queries can be used to task based on specific requirements
  + You can also visualize these using charts
  + You can save the query as a my query or a shared query
  + Then you can use the query as and when needed without having to create that again
* Adding queries chart to Dashboard:
  + To add the chart of query to dashboard you need to save the query as shared query.
* Permissions:
  + You can control the permissions at project and organization levels.
  + Organization Settings> Permissions
  + Project Settings> Permissions
* Query Permissions:
  + All users except members of Readers group can create and edit their own queries.
  + To edit queries under shared queries user must be part of Project administrators group.
  + For saving under shared queries, user must have basic access level or higher.
  + However, you can also specify permissions at the specific query without exposing other resources.
  + Steps:

Go to shared query>security>users>set the required permissions>save

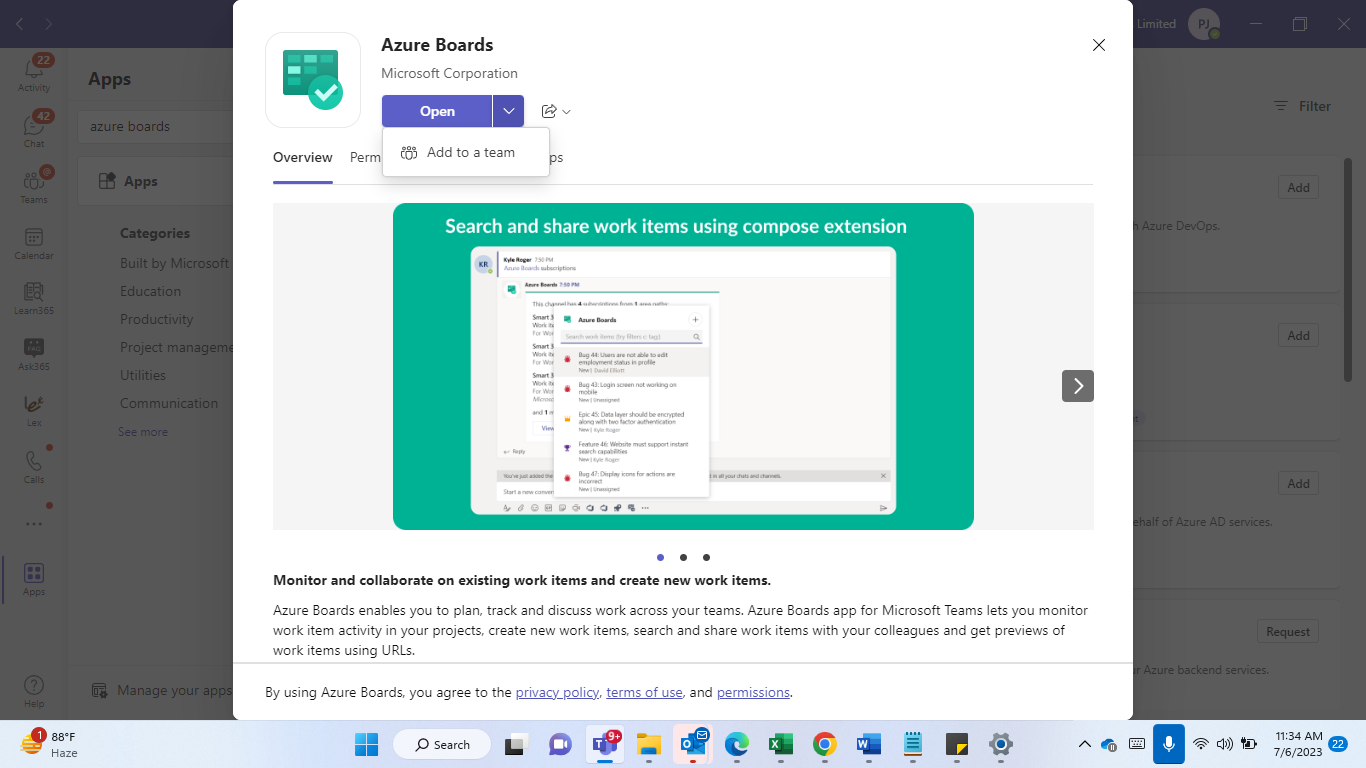
* Wiki
* Delivery Plans
* Link Teams to Azure Boards:
  + Go to Team>Applications>Azure Boards>Install/Add
  + After installation go to azure boards and sign in using azure credentials using below command.

@azure boards signin

* + Once signed in link the azure devops project.

@azure boards link Project URL

* + For convenience map the azure boards to a teams created specifically for that project tracking.



* + After linking you can subscribe to specific events based on your requirements using add subscription button. This will notify you over teams as when that event occurs.
* Source Control Management:
* Git:
  + Install Git
  + Create a new folder/directory
  + Initialize the above created directory by using below command.

git init

* + Initialization of git creates a new directory with name .git which acts as git’s database
  + Configure username and password using below command.

git config --global user.name “Pravin Jangid”

git config --global user.email “[pravinjangid94@gmail.com](mailto:pravinjangid94@gmail.com)”

* + To move the file to staging directory use below command.  
    git add . : applicable to all files in current directory

git add filename : applicable to specific files in current directory

* + To commit the changes and move the file to git directory use below command

git commit -m “Your message”

* + To check the status of files to be committed use below command

git status

* + After committing the file after each modification you save you will see an M symbol next to the file name.
  + To get rid of this and sync the changes you again need to follow the earlier process.

git status : Indicates the modification

git add filea.txt : moves the file to staging directory

git commit -m “Message” : commits the changes

* + Follow below process to move to some earlier commits.

1. Use git log to get the details of commits.
2. Use git checkout commitid to switch to specific version of the file.

Example. git log

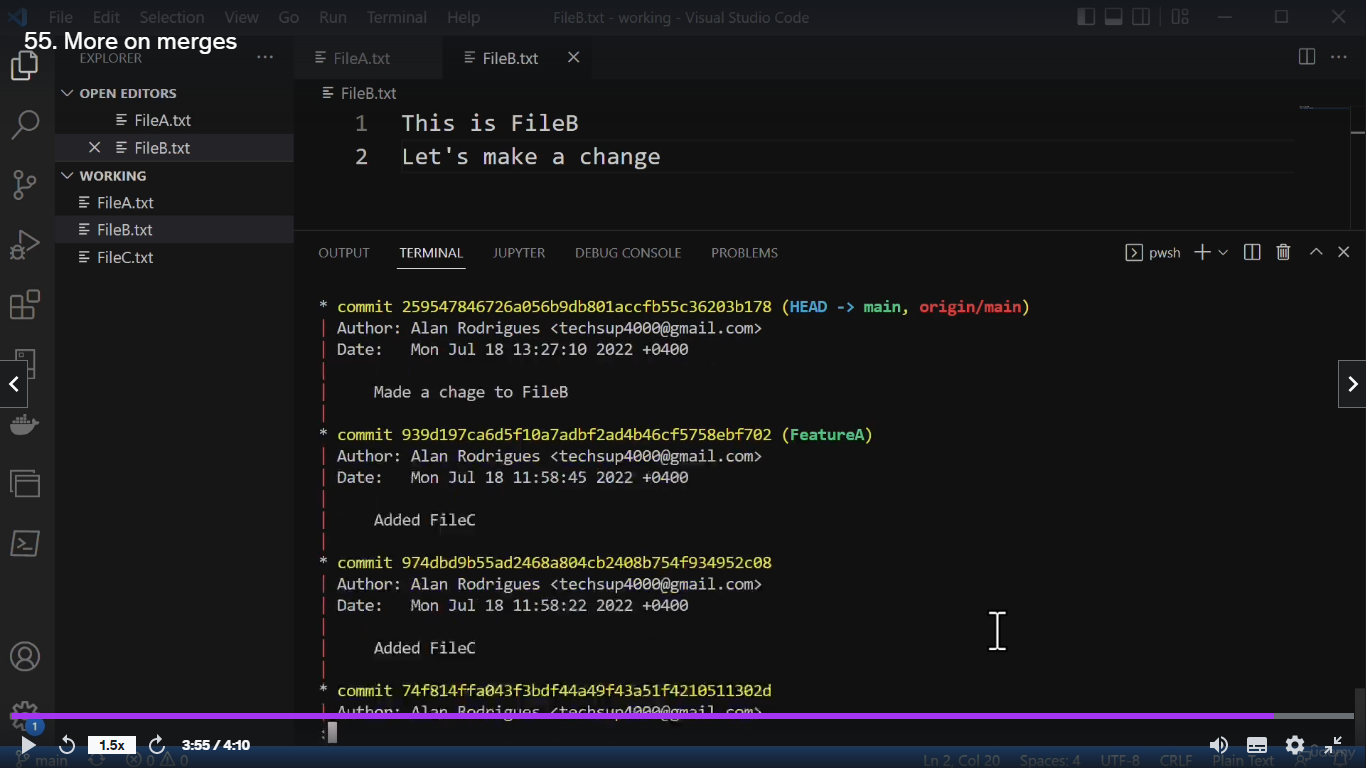
This will display all the commits made till the current version.

git checkout abcd1234

This will switch to the version which was committed as part of abcd1234 commit

* + After jumping back to abcd1234 commit version you won’t be able to see the commits made after that version. To see the future commits made after abcd1234 use git log --all

git log --graph : displays log history in detailed view as below:



* + From git log --all you will see the future commits and then again come to future commits using git checkout commitid
* Rollback the file from staging area:
  + Use git rm --cached filea.txt
  + git status : This will show as untracked file.
  + Revert the changes and save again.
  + git status : This will show as untracked file.
  + git add filea.txt : Adds the original file to staging area
  + git status : This will show nothing to commit.
* Branches:
  + To create a new branch use below command.

git checkout -b BranchName

git checkout -b FeatureA

* + Branches can also be created using below commands.

git branch featureA

* + To merge the branch back to main follow below process.
    - Switch to main branch using git checkout main
    - After moving to main branch run below command.

git merge branchname

git merge FeatureA

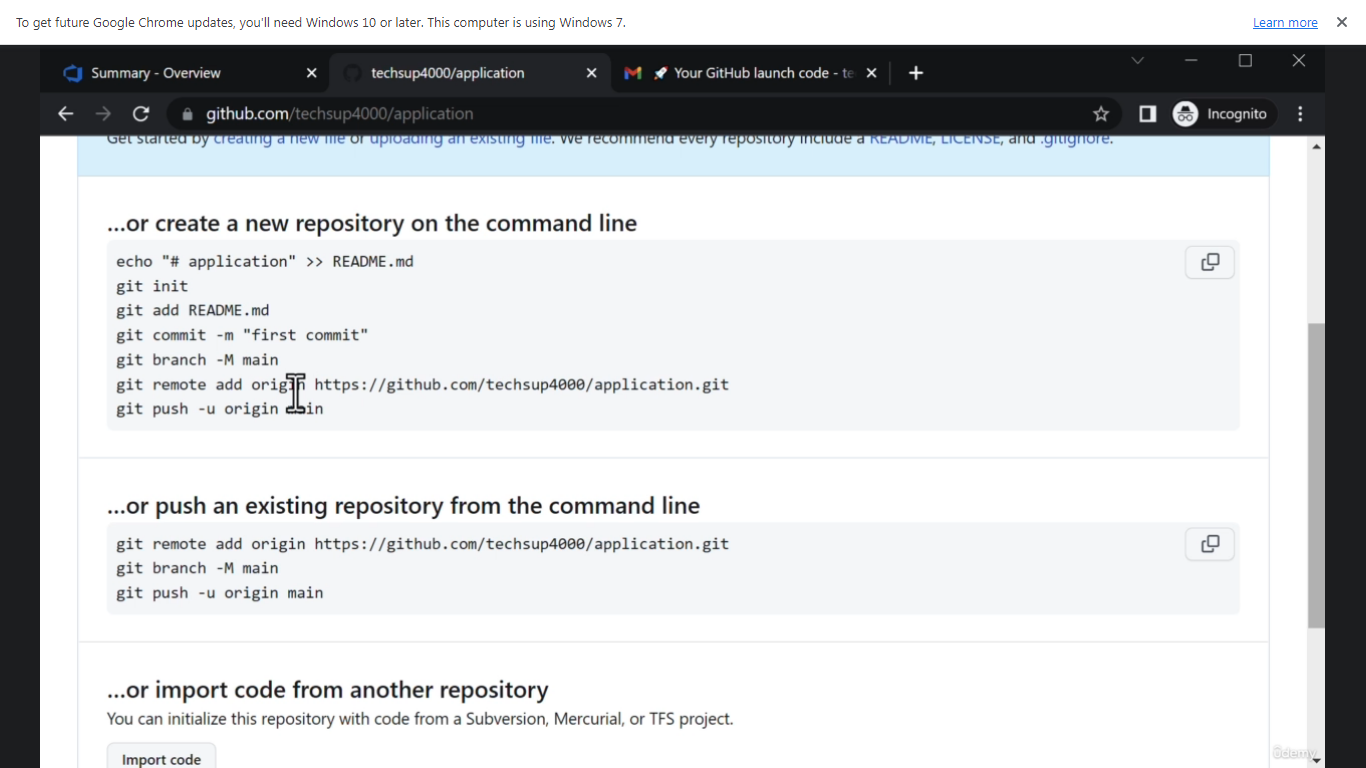
* + - After the merge you will see all the files that were in FeatureA in main as well.
* Create Github account:
* Add GitHub Repository:
  + Create a new github repository
  + Add remote repository to git using below command:

git remote add origin RepositoryURL

git remote add origin <https://github.com/techsup4000/application-git>

My repository:DevOpsAZ400

git remote add origin https://github.com/pravinjangid94/DevOpsAZ400.git

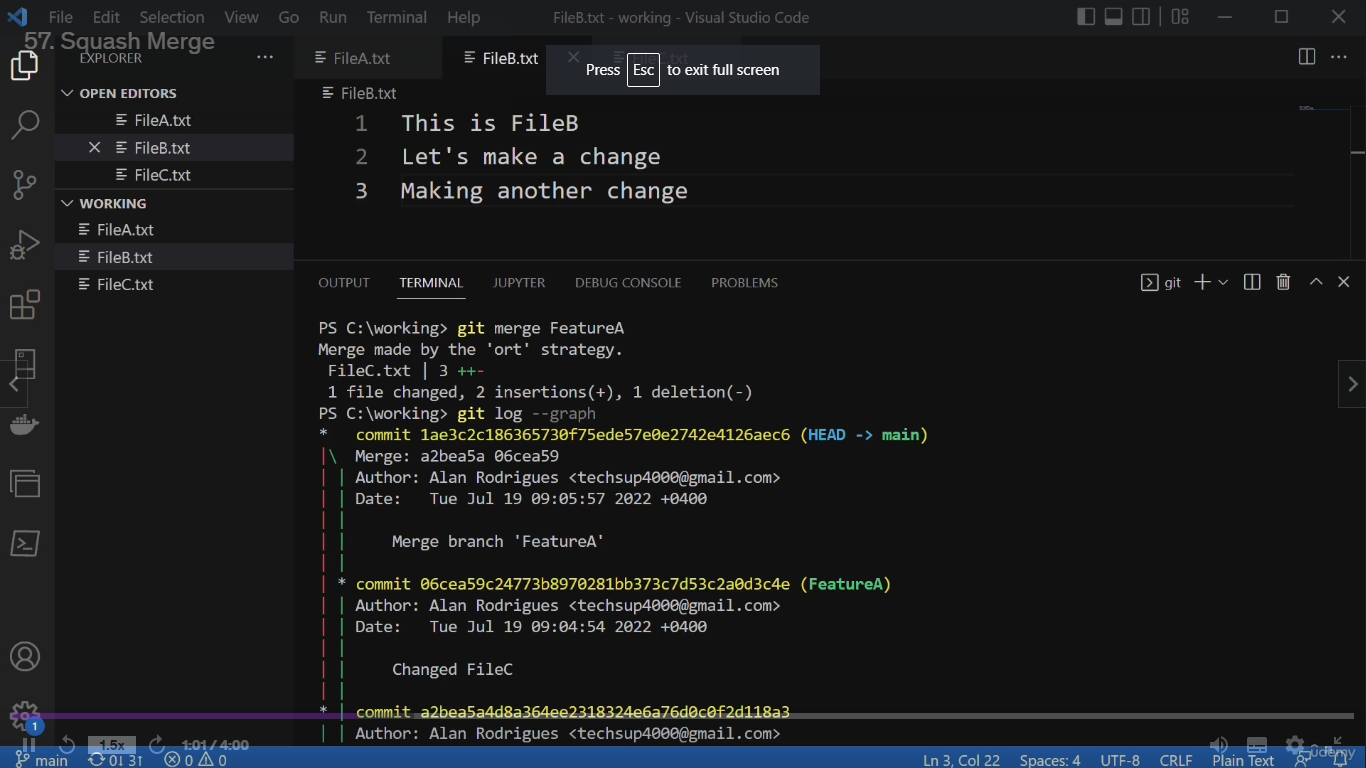


Then push the files from local repository to remote repository using below command.

git push –u origin main

this will prompt to login using your remote repository credentials.

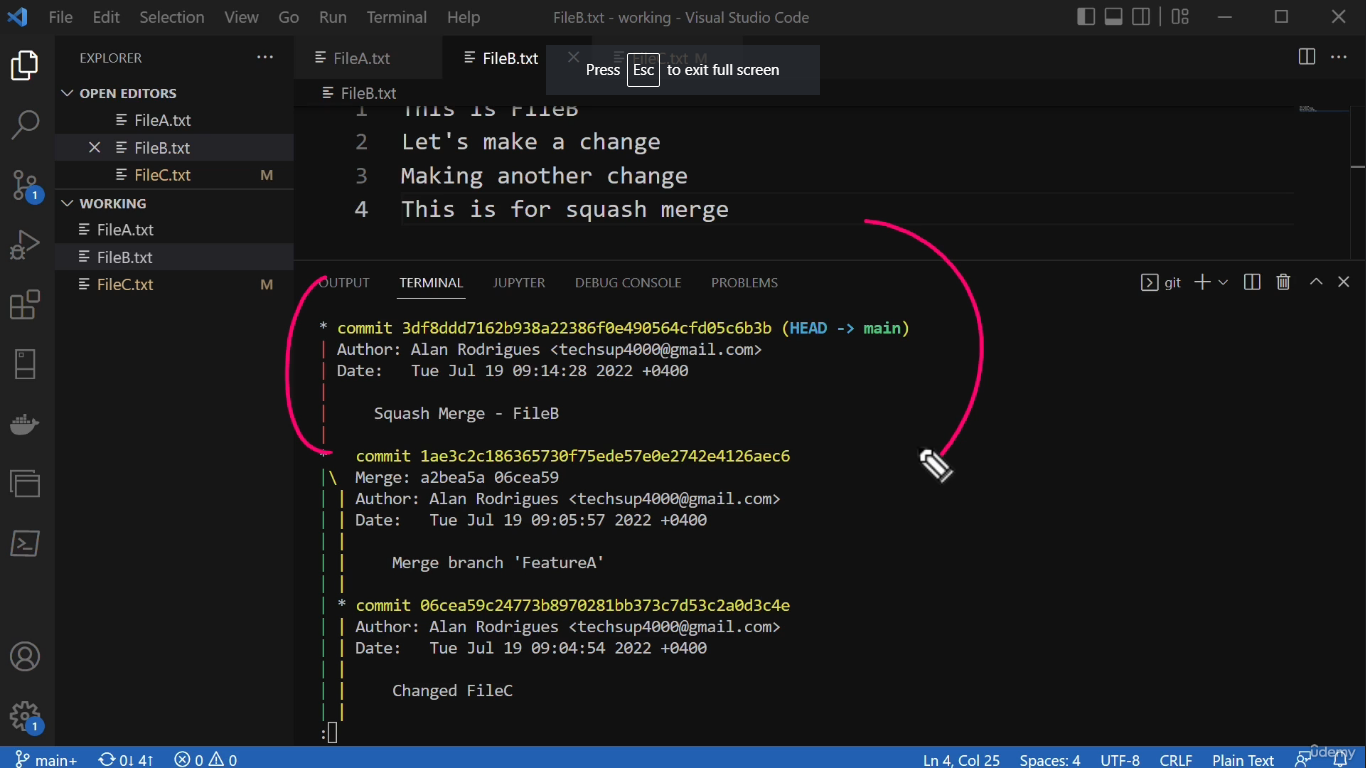
* + Your changes made on local repository will reflect on remote repository only and only when you push them using git push –u origin main command
* Merges:
  + Fast-forward merge: When the change is made only on the feature branch, git prefers fast-forward merge. There must be no change to the main branch
  + ORT strategy: When after the creation of a feature branch changes are carried out on both main and feature branch, git uses ORT strategy. As part of this git first commits the changes on main branch (this is one commit), then changes on feature branch (this is second commit) and finally creates a commit for merge as well.



* + Squash Merge: git merge –squash featurea to get rid of so many commit messages above you can use squash mere. For example, use the below command to perform a merge. After the merge if you run the git log --graph command you will only see a single commit message for the merge unlike above scenario where there was a separate commit message for each merge and then final merge as well.
  + git push –u origin --all

Pushes all the branches to origin. If you want to push only changes made to specific branch, run the following command.

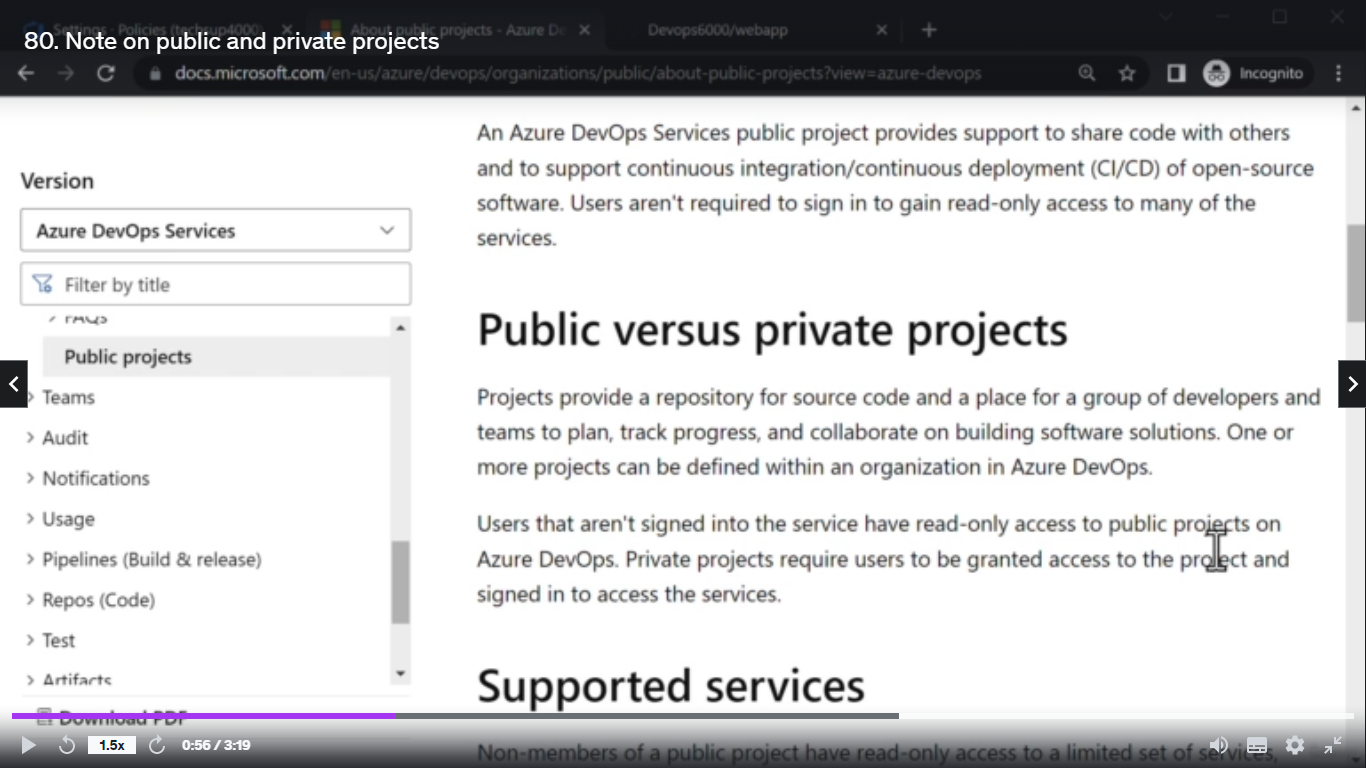
git push –u origin main This will push only the changes made to main branch.



* Azure Repos Integration with MS Teams:

Similar to Azure Boards integration

* Integration of Azure Boards with Github:
  + To integrate the azure boards with github, go to github marketplace and install the azure boards from there.
  + Configure the required repositories.
  + To link the work items from azure boards enter the comments as AB#TaskID. For example, if you wish to link the work item that has ID 18 in azure boards in the comments of github pull requests enter the comment as AB#18. This is create a link between work item and the pull request.
* Team Foundation Version Control:
  + Alternative of Git. It is different version control system. Not so popular.
* Public vs Private Projects:
  + By default, public projects are disabled on azure repos.
  + To enable public projects, navigate to organization settings > Policie(under security). Enable allow use of public projects.



* Cherry Picking Changes:
  + If you wish to merge only changes made part of some specific commit you could do that using below command.

git cherry-pick commidid

here commitid is the commit id of the commit that you wish to cherry-pick

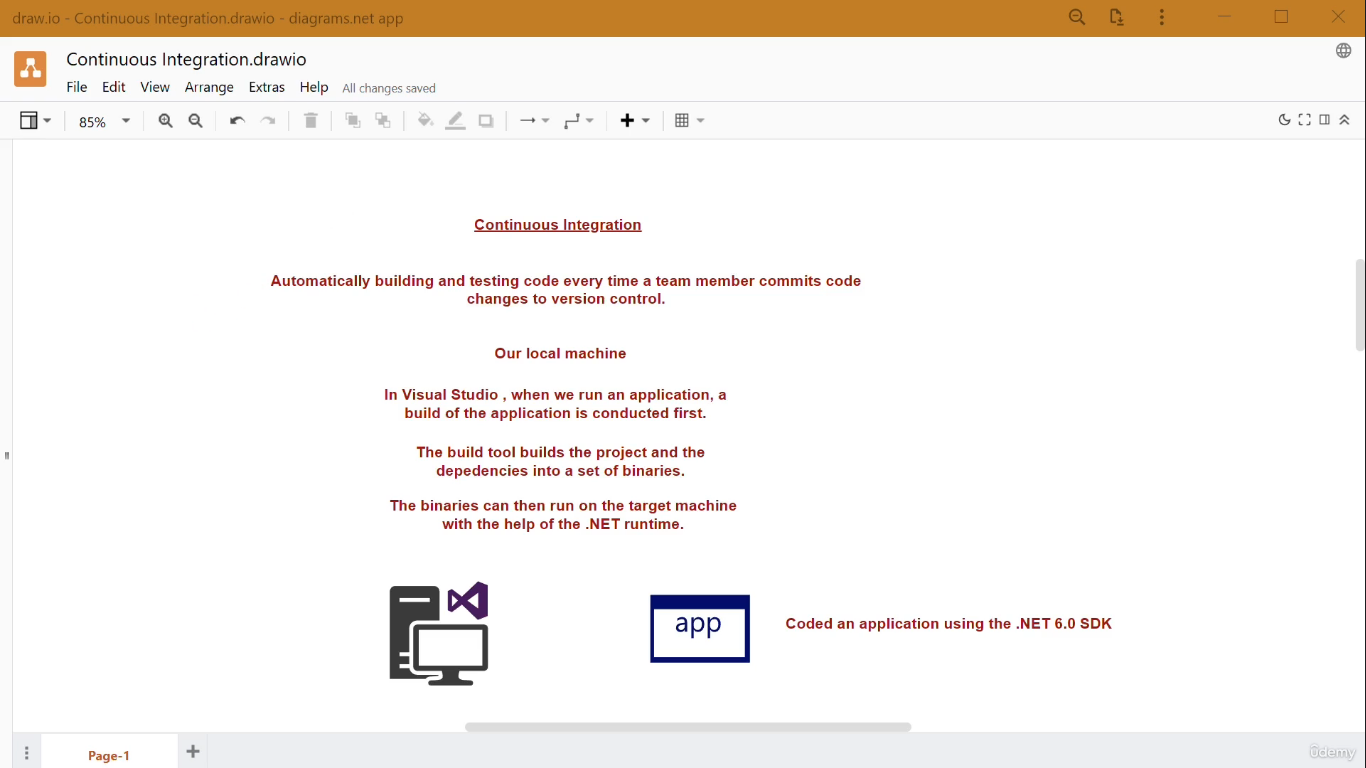
* + Example, you created feature branch out of main branch and made 2 changes ChangeA and ChangeB to it. However, you wish only to merge the changes made as part of ChangeA to the main branch then you can run below command.

git cherry-pick ChangeACommitID

ChangeACommitID represents the commit ID for the commit of Change A.

**Build Pipelines**

* Continuous Integration:



* Continuous Integration is automatic building and testing code every time a team member commits code changes to version control.
* Steps for creation of basic build pipeline:
  + Pipelines>Create Pipeline>Where is your code hosted(Azure repos)> Select the repository>Select the underlying programming language(ASP.Net Core).
  + This will create a default YAML definition file.
  + Components:
    - Trigger : branch which should act as a trigger
    - Pool : VM Image of Microsoft hosted agents
    - Variables : Name value pairs that can be used across this pipeline
    - Steps : tasks to be performed as part of this pipeline

* + Modify and make sure that only the required steps are there and then click on save and run.
  + The YAML code is also version controlled and hence will be a part of the repository.
  + A job starts to perform the tasks specified in above pipeline.
  + Click on job to track the progress of all the tasks.
  + During the build pipeline the code is copied to Microsoft hosted agents or self hosted agents and then build process is performed on those agents.
  + In case of MS hosted agents the VMs are destroyed once the build pipeline is complete/finished.
* Modify the code in your repo to see whether and how the build pipeline gets triggered
* Self-Hosted Agent:
  + Create a VM which you wish use as a SHA.
  + Perform below steps on the SHA to configure it for use:
    - Login to Azure portal, generate PAT and paste it in notepad for future use.
    - Go to Organization Settings>Agent Pools (under Pipelines)>Select the required pool(Default for this example)
    - Click on New agent, download the required agent file and follow the instructions.
    - Extract the contents of agent file and run config.cmd file to initiate the configuration by browsing to that location through cmd and running ./config.cmd
    - On the configuration page specify following details:
    - Enter Server URL: DevOps Organizaton URL.
    - Authentication type: PAT : yes
    - Enter the PAT copied above.
    - Agent Pool
    - Agent Name
    - Enter work folder:
    - Run agent as a service?:
    - Enter other such more details and refresh the agent page.
    - You should see the agent as online.
    - Next, to use the above self hosted agent for running the build pipeline make changes to the pool section of the pipeline.
    - In the pool section delete the vmImage and replace it by below:

name: Default

Here, we have written Default because we created the agent in Default pool. In case, your agent is available in other pool you have to specify the name of that pool.

pool:

name: PoolName

* Understanding how the pipeline works on SHA:
  + Removing the task of .net and NuGet as .net is already installed and NuGet is not needed for the build of above example.
  + By Default, zipAfterPublish is true. If you don’t want the file to be compressed then make sure to add it and mark it as false as below.

zipAfterPublish: false

* + Build.ArtifactStagingDirectory: We use to publish the build at this predefined variable. The value of this predefined variable is C:\working directory\1\a
  + Hence, in this example the build is stored at above path.
* Analytics of Pipelines are available at Pipelines>Select the pipeline for which you want to see the analytics data> Select Analytics
* Also, as your azure-pipelines.yml is getting version controlled, you can see all the history of that file and switch back to some if needed.
* **Security:**
* Security is important at every stage whether it be planning, development, etc.
* Mend(WhiteSource)
  + is an open source security and compliance management tool.
  + Checks for security vulnerabilities and licensing issues
  + As a prerequisite you must have Azure AD linked to your DevOps organization. To verify, whether it is linked with AAD navigate to AAD.
  + Post this, go to Extensions>Browse Marketplace>search for Mend>select Mend Bolt>Install/Add
  + Mend will now appear under extensions(organization settings). Click on it and create your mend account.
  + After the configuration of Mend Bolt, create a sample project and push it to the newly created organization(AAD linked). Once the build is complete, click on pipeline>select the latest run. You should see an additional tab name Mend. This tab will have a report of quality.
  + Now, add a nuget package and then push again. Browse to see the report to the mend for the latest run.
* Deleting the newly created organization:
  + Naviagate to organization settings> Overview at the bottom you will see the option to delete the organization. We are deleting this as this was only created for the demo of mend bolt.
  + To sign in back to the previous organization login using Microsoft account.
* Unit Tests:
  + Unit test should be defined as part of code.
  + Add below command to pipeline to make sure that unit tests specified in the project code are run.

- task: DotNetCoreCLI@2

inputs:

commands: test

projects: ‘\*\*./webtest.csproj” #(name must be same as project speicified in code)

arguments: ‘--configuration $(buildConfiguration)’

* + After running the pipeline a new tab names Tests will appear when you select the run.
* Code Coverage:
  + Similar to unit test, the first step is to have a unit test in place in the code that tests some part of the code.
  + In this example, we have modified the code to add 2 numbers and the webtest project tests whether the obtained sum is correct.