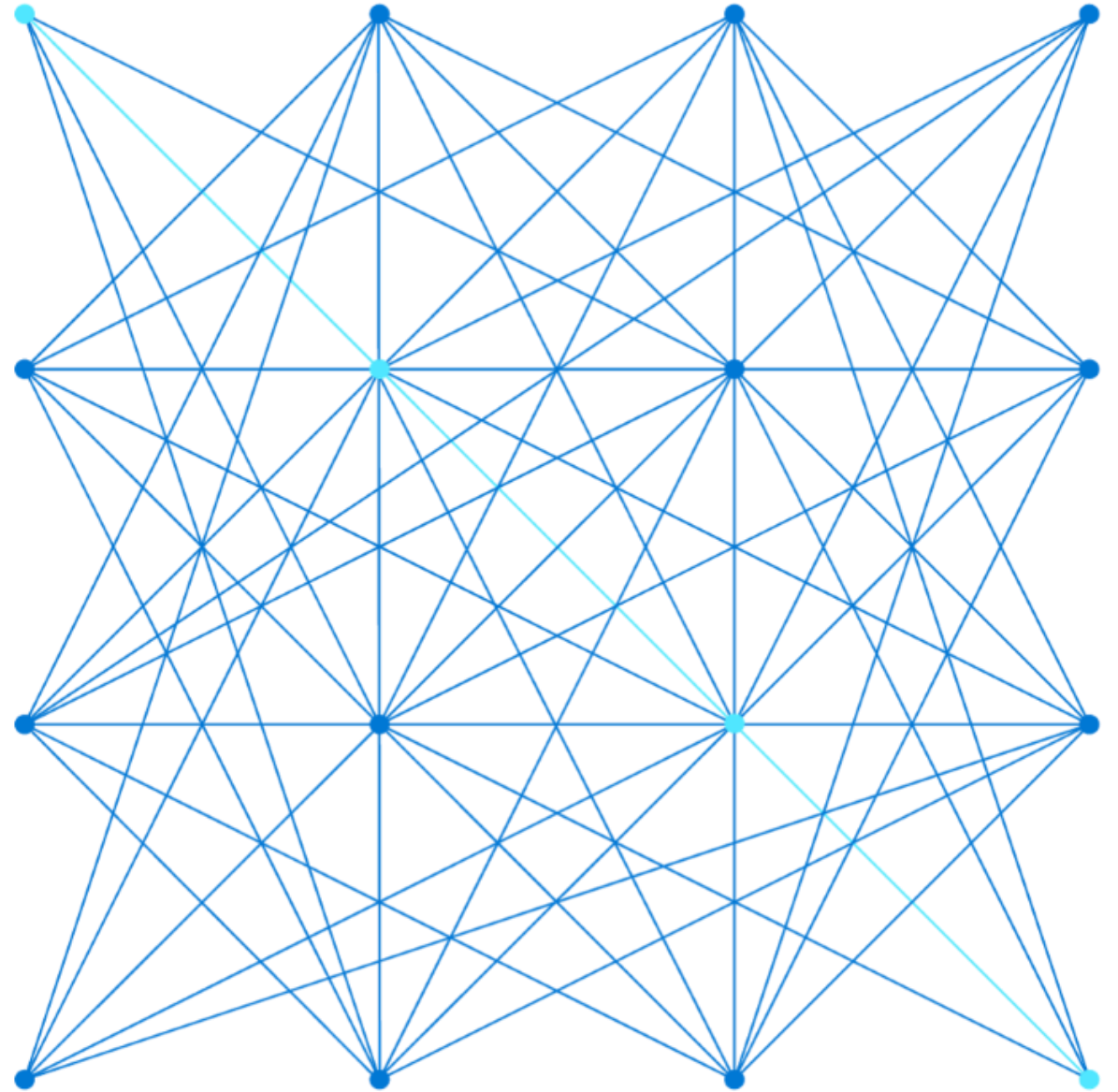


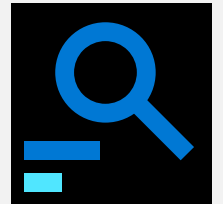
AZ-400.00

Module 2:

Getting Started with Source Control



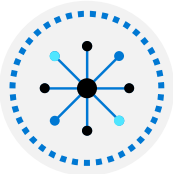
Lesson 01: Module overview



Module overview



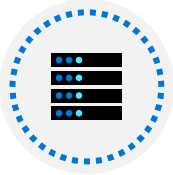
Lesson 1: Module Overview



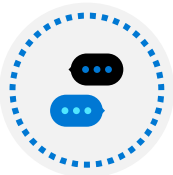
Lesson 2: What is source control?



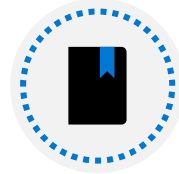
Lesson 3: Benefits of source control



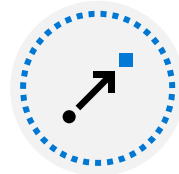
Lesson 4: Types of source control systems



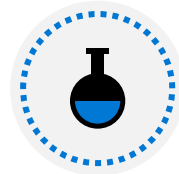
Lesson 5: Introduction to Azure Repos



Lesson 6: Introduction to GitHub



Lesson 7: Migrating from Team Foundation Version Control (TFVC) to Git in Azure Repos



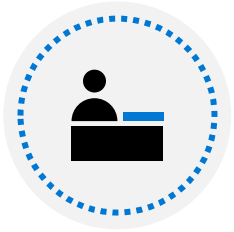
Lesson 8: Lab



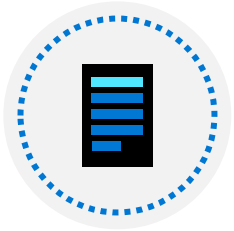
Lesson 9: Module review and takeaways

Learning objectives

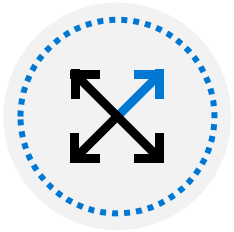
After completing this module, students will be able to:



Describe the benefits of using Source Control



Describe Azure Repos and GitHub



Migrate from TFVC to Git

Lesson 02: What is source control?



(#mw) Introduction to source control

When it was introduced, DevOps was a revolutionary way to release software quickly and efficiently while maintaining a high level of security.

Source control (version control) is a critical part of DevOps.

Foundational practices of DevOps

Foundational practices and the stages of DevOps evolution: stages 0 to 2

	Defining practices and associated practices	Practices that contribute to success
Stage 0	• Monitoring and alerting are configurable by the team operating the service • Deployment patterns for building applications or services are reused • Testing patterns for building applications or services are reused • Teams contribute improvements to tooling provided by other teams • Configurations are managed by a configuration management tool	
Stage 1	• Application development teams use version control • Teams deploy on a standard set of operating systems	• Build on a standard set of technology • Put application configurations in version control • Test infrastructure changes before deploying to production • Source code is available to other teams
Stage 2	• Build on a standard set of technologies • Teams deploy on a single standard operating system	• Deployment patterns for building applications and services are reused • Rearchitected applications based on business needs • Put system configurations in version control

* The practices that define each stage are highlighted in bold font

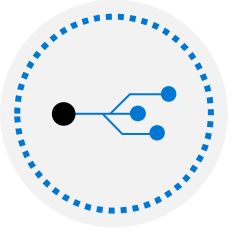
Foundational practices of DevOps

Foundational practices and the stages of DevOps evolution: stages 3 to 5

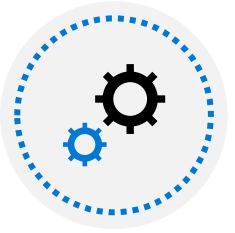
	Defining practices and associated practices	Practices that contribute to success
Stage 3	• Individuals can do work without manual approval from outside the team • Deployment patterns for building applications and services are reused • Infrastructure changes are tested before deploying to production	• Individuals can make changes without significant wait times • Service changes can be made during business hours • Post-incident reviews and results are shared • Teams build on a standard set of technologies • Teams use continuous integration • Infrastructure teams use version control
Stage 4	• System configurations are automated • Provisioning is automated • Application configurations are in version control • Infrastructure teams use version control	• Security policy configurations are automated • Resources made available via self-service
Stage 5	• Incident responses are automated • Resources available via self-service • Rearchitected applications based on business needs • Security teams are involved in technology design and deployment	• Security policy configurations are automated • Application developers deploy testing environments on their own • Success metrics for projects are visible • Provisioning is automated

* The practices that define each stage are highlighted in bold font

What is source control?



A source control system allows developers to **collaborate on code and track changes**.



Source Control Management (SCM) systems provide a **running history** of code development and resolve conflicts when merging contributions from multiple sources.



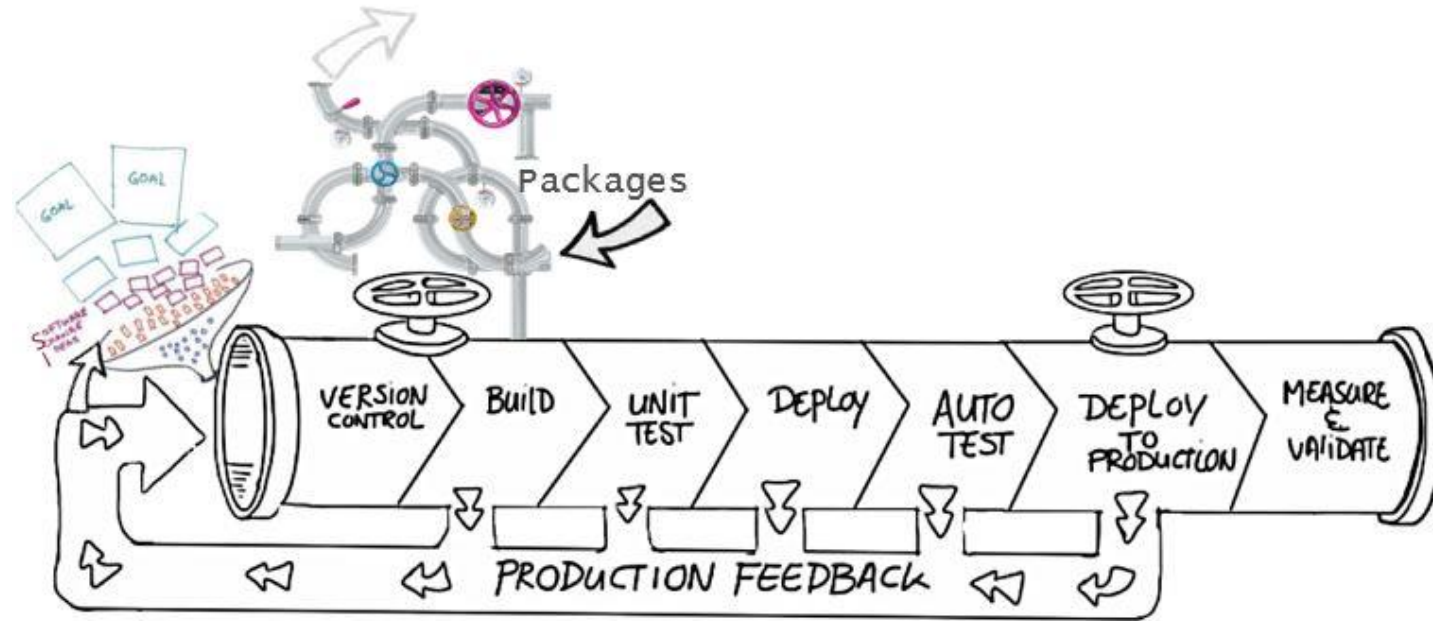
Source control protects source code from both catastrophe and the casual degradation of human error and unintended consequences.

Lesson 03: Benefits of source control



Benefits of source control

(#mw) from your notes, benefits: Reusability, traceability, manageability, efficiency, collaboration, and learning



Create workflows

Work with versions

Collaboration

Maintains history of changes

Automate tasks

Best practices for source control



Make small changes



Don't commit personal files



Update often and right before pushing to avoid merge conflicts



Verify your code change before pushing it to a repository; ensure it compiles and tests are passing



Pay close attention to commit messages as these will tell you why a change was made (#mw: [see](#))

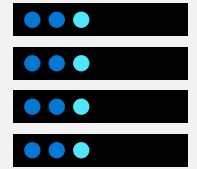


Link code changes to work items



No matter your background or preferences, be a team player and follow agreed conventions and workflows

Lesson 04: Types of source control systems



Centralized source control



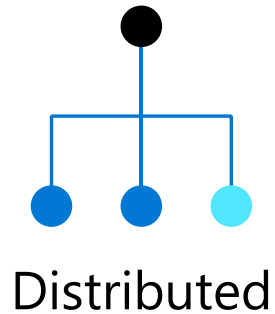
Centralized

Strengths	Best Used for
• Easily scales for very large codebases • Granular permission control • Permits monitoring of usage • Allows exclusive file locking	• Large integrated codebases • Audit and access control down to the file level • Hard to merge file types

There is a single central copy of your project, and programmers commit their changes to this central copy.

Common centralized source control systems are TFVC, CVS, Subversion (or SVN), and Perforce.

Distributed source control



Strengths

- Cross platform support
- An open-source friendly code review model via pull requests
- Complete offline support
- Portable history
- An enthusiastic growing user base

Best Used for

- Smaller size (in bytes) and modular codebases
- Evolving through open-source
- Highly distributed teams
- Teams working across platforms
- Greenfield codebases

Every developer clones a copy of a repository and has the full history of the project.

Common distributed source control systems are Mercurial, Git, and Bazaar.

Git and TFVC

Git:

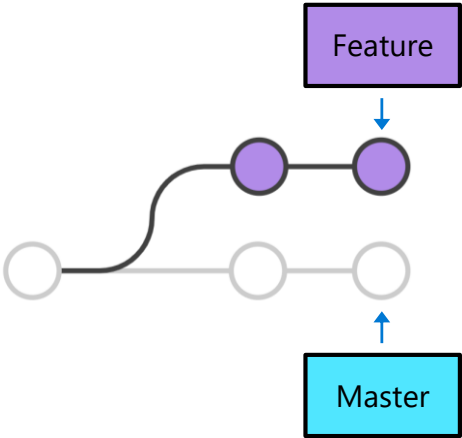
- Distributed Source Control system
- Each developer has a copy of the source repository on their development system

TFVC:

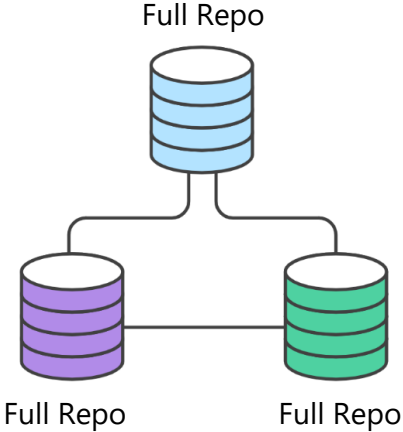
- Centralized Source Control system
- Team members have only one version of each file on their dev machines.
- In the **server workspaces** model, before making changes, team members publicly check out files.
- In the **local workspaces** model, each team member takes a copy of the latest version of the codebase with them and works offline as needed.

Why Git?

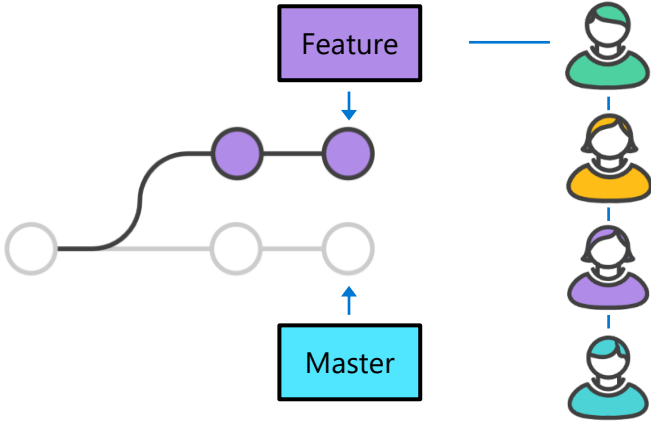
Feature branches



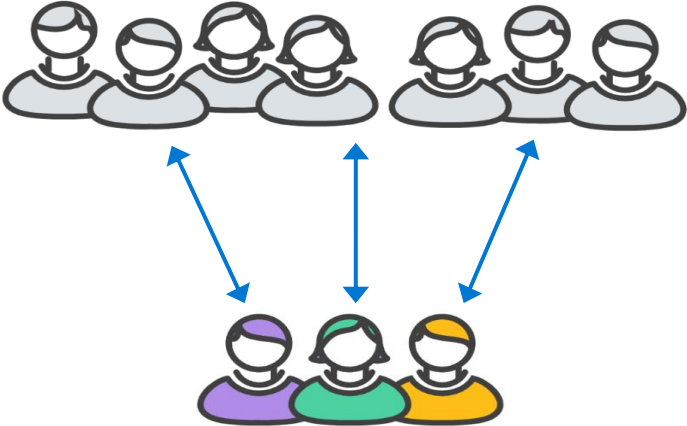
Distributed development



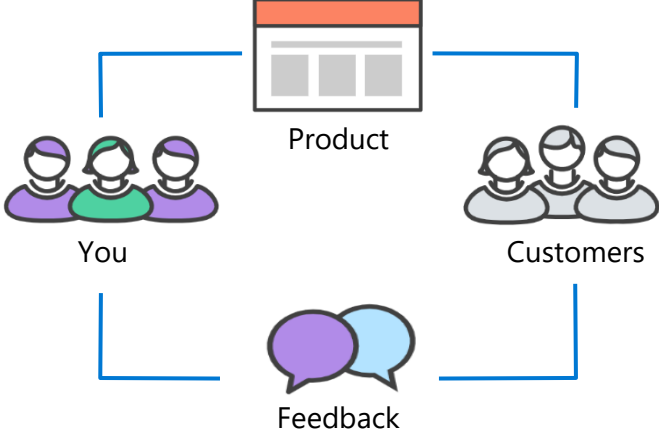
Pull requests



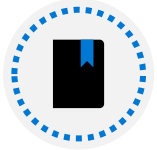
Community



Release cycles



Objections to using Git



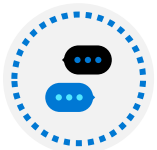
Overwriting History – If used incorrectly, it can lead to conflicts (#mw: see [here](#))



Large Files – Git works best with repos that are small and that do not contain large files (or binaries); consider using Git LFS support (#mw: see [here](#). More on this later.)

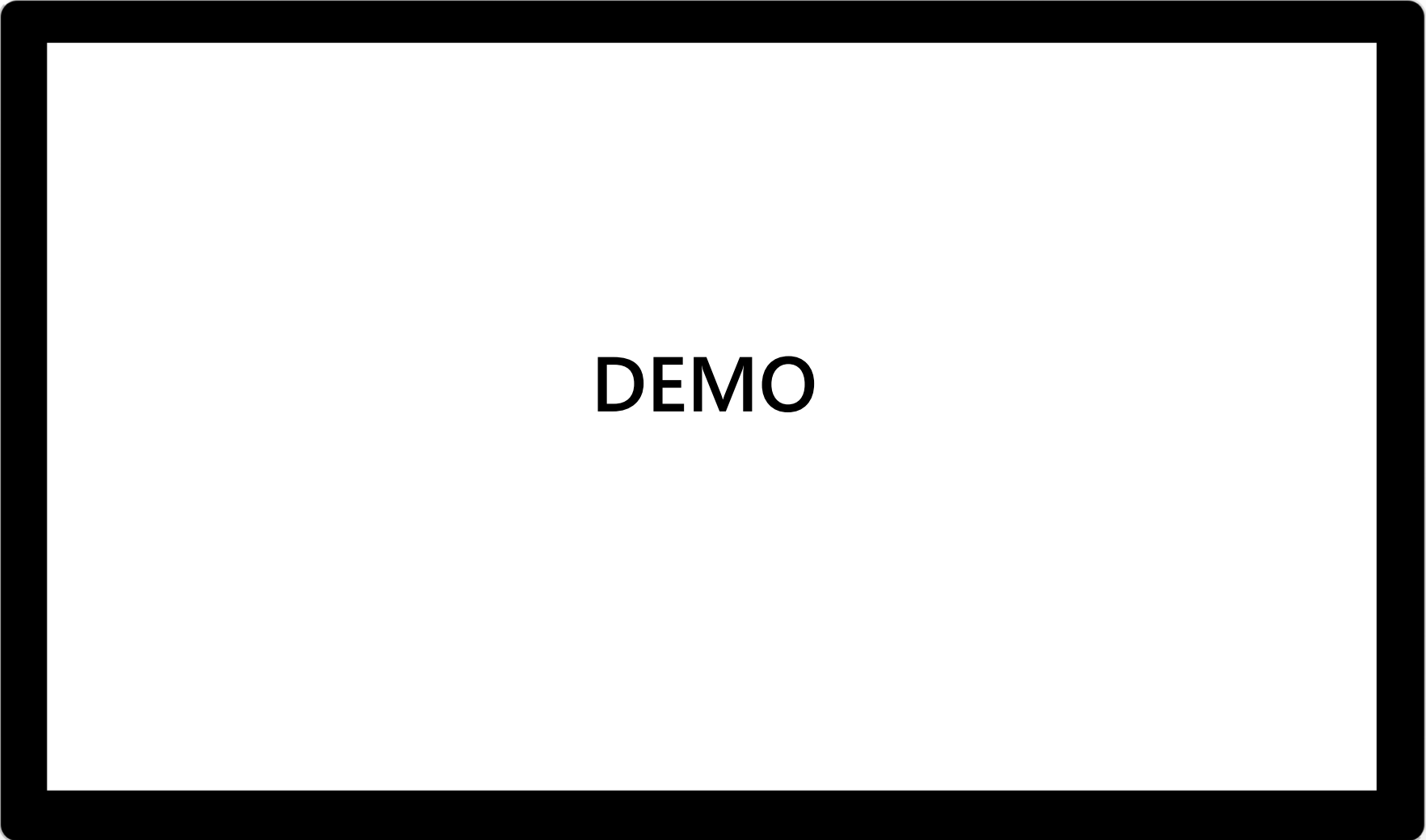


Learning Curve – Some training and instruction will be needed. Existing developers might need to be retrained.



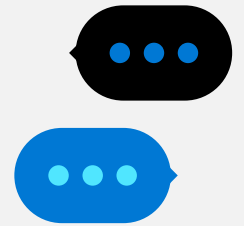
Discussion – What objections do you have?

Working with Git locally

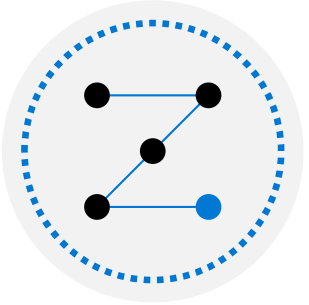


DEMO

Lesson 05: Introduction to Azure Repos



Azure Repos



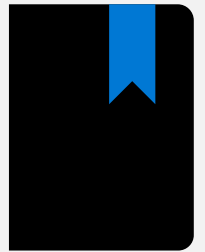
Set of version control tools to manage your code:

- Tightly integrated with other Azure DevOps features
- Can connect from common development environments
- Policy based management to protect branches
- Offers two styles of version control:

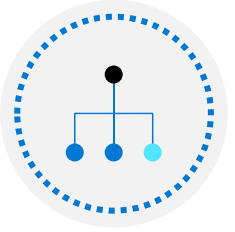
Git: Distributed version control

Team Foundation Version Control (TFVC): Centralized version control

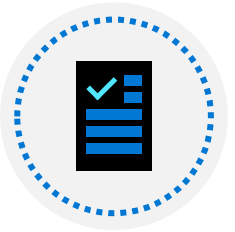
Lesson 06: Introduction to GitHub



What is GitHub ?



Largest open-source community



Features of GitHub:

- Automate from code to cloud
- Securing software, together
- Seamless code review
- Code and documentation in one place
- Coordinate
- Manage teams

Linking GitHub to Azure Boards

Azure Boards App links commits, pull requests, and issues, directly to work items

Authenticate to GitHub using a username/password or a personal access token (PAT)

Marketplace / Apps / Azure Boards



Application

Azure Boards

[Set up a plan](#)

 Verified by GitHub
GitHub confirms that this app meets the [requirements for verification](#).

Categories

[Project management](#)

[Recently added](#) [Free](#)

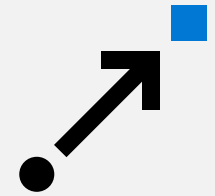
Plan, track, and discuss work across your teams

Azure Boards offers Kanban boards, backlogs, and dashboards for flexible work tracking that is fully connected to the code and issues for all your projects – big and small.

A separate [Azure DevOps Services subscription](#) is required. We'll help you get started for free if you don't already have one.

(#mw) See [here](#) and [here](#)

Lesson 07: Migrating from Team Foundation Version Control (TFVC) to Git in Azure Repos



Migrating from TFVC to Git

(#mw: [I found this useful](#))

Single branch
import

Full synchronization
(git-tfs)

PartsUnlimited

PartsUnlimited ▾ Files

Favorites All repositories

Filter repositories

PartsUnlimited ☆

+ New repository

↑ Import repository

⚙ Manage repositories

Import from TFVC

Source type

TFVC ▾

ⓘ Migrating from TFVC to Git can be disruptive. Before starting the import, we suggest reading our [documentation](#)

Path *

e.g. \$/Contoso/HelloWorld

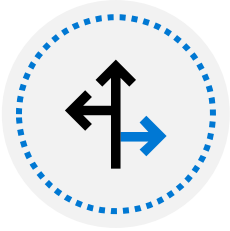
☒ Migrate History

For 180 days

Name *

Name your new Git repository.

Migrating from TFVC to Git



Migrating the tip:

- Only the latest version of the code
 - History remains on the old server
-



Migrating with history:

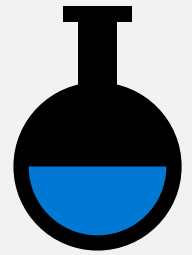
Tries to mimic the history in Git



Recommend to migrate the tip, because:

- History is stored differently – TFVC stores change sets, Git stores snapshots of the repository
- Branches are stored differently – TFVC branches folders, Git branches the entire repository

Lesson 09: Lab



Lab: Version controlling with Git in Azure Repos



Lab overview:

In this lab you will establish and work with a local Git repository.

Objectives:

Clone an existing repository

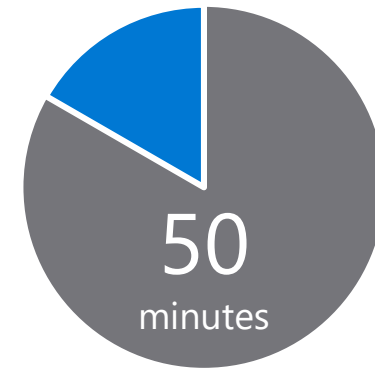
Save work with commits

Review history of changes

Work with branches by using Visual Studio



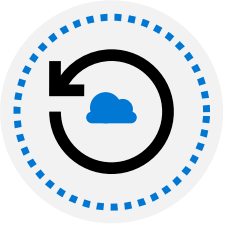
Duration:



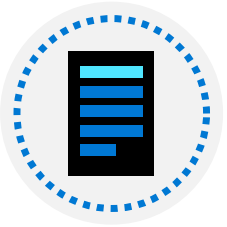
Lesson 10: Module review and takeaways



What did you learn?



Describe the benefits of using source control



Describe Azure Repos and GitHub



Migrate from TFVC to Git

Module 2 review questions

1

What are some of the benefits of source control?

2

What are the benefits of using distributed version control?

3

What are the benefits of using centralized version control?

4

What is source control?