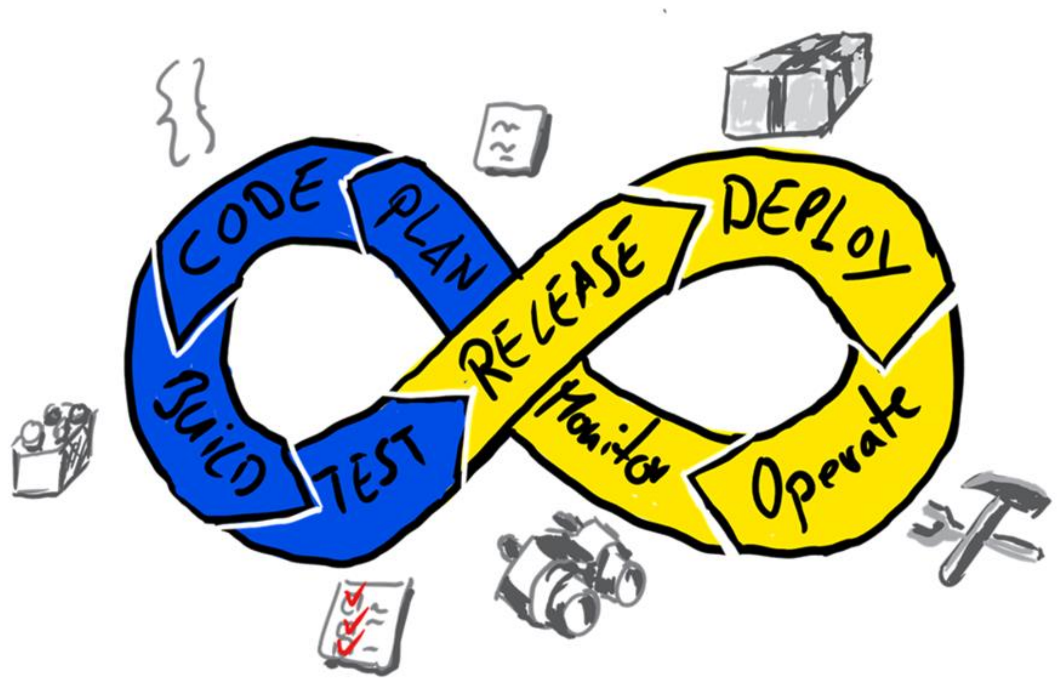
Microsoft

Module 1: Getting Started with Source Control

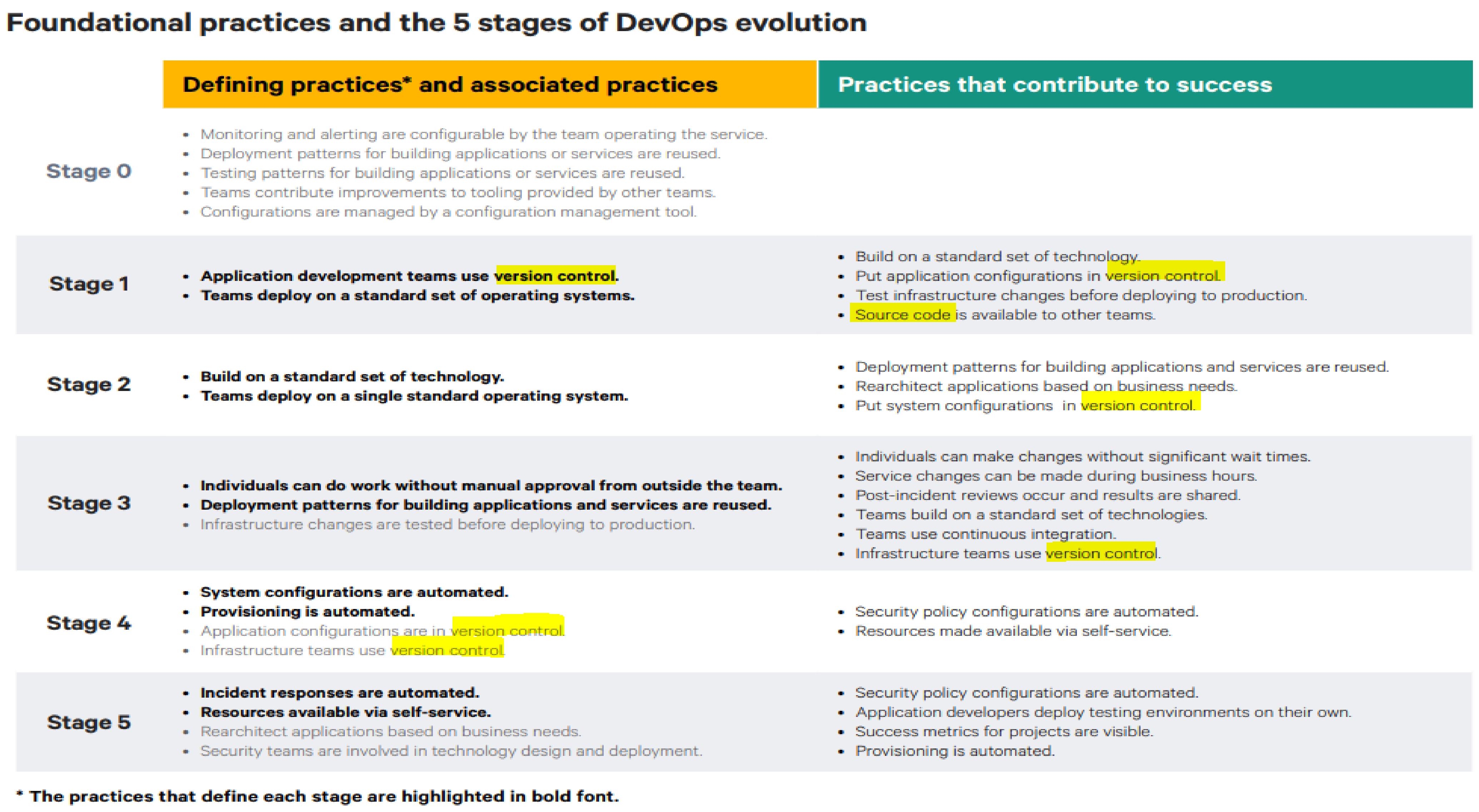
Lesson 01: What is Source Control?

Introduction to Source Control

- DevOps is 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 5 stages of DevOps evolution

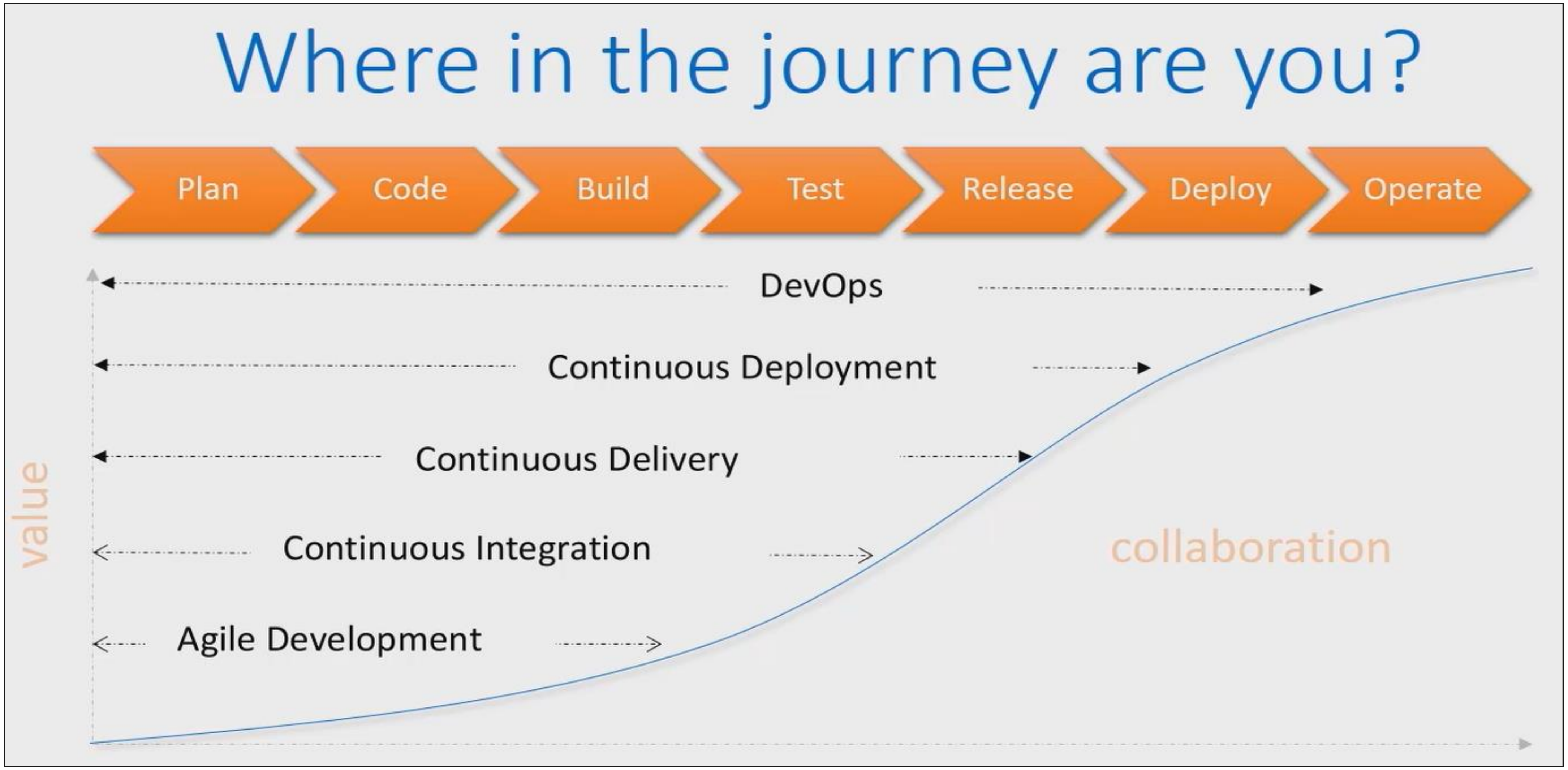
	Defining practices* and associated practices	Practices that contribute to success
Stage 0	• Monitoring and alerting are configurable by the team operating the service. • Deployment patterns for building applications or services are reused. • Testing patterns for building applications or services are 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 technology.** • **Teams deploy on a single standard operating system.**	• Deployment patterns for building applications and services are reused. • Rearchitect applications based on business needs. • Put system configurations in version control.
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 occur 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.** • Rearchitect 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?

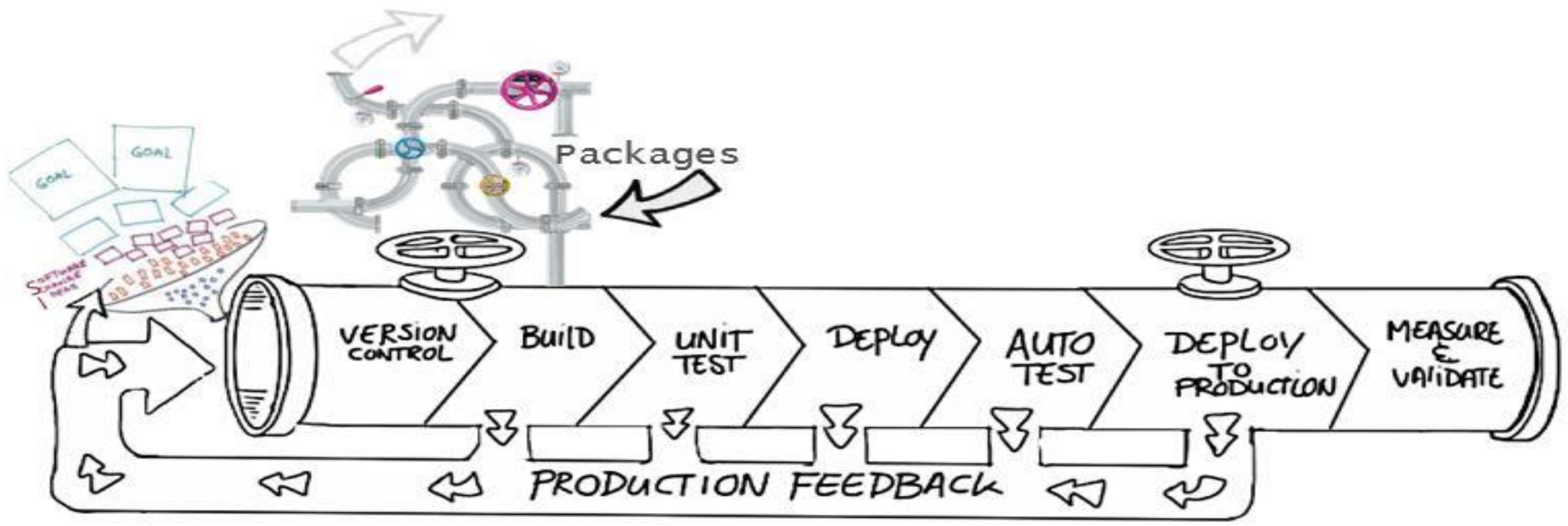
- Source control is the practice of tracking and managing changes to code
- Source control management (SCM) systems provide a running history of code development and help to resolve conflicts when merging contributions from multiple sources
- Source control protects source code from both catastrophe and the casual degradation of human error and unintended consequences
- Benefits include: reusability, traceability, manageability, efficiency, collaboration, and learning.

Introduction to DevOps



Lesson 02: Benefits of Source Control

Benefits of Source Control

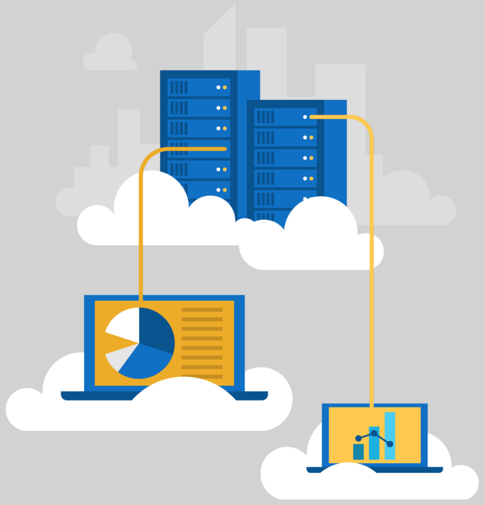


- Create workflows
- Work with versions
- Collaboration
- Maintains history of changes
- Automate tasks

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
- Link code changes to work items
- No matter your background or preferences, be a team player and follow agreed conventions and workflows

Lesson 03: Types of Source Control Systems



Introduction to Source Control

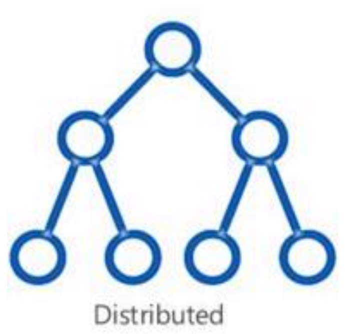
- The DevOps pipeline
- Why Source Control?
- Source Control Systems
- Centralized vs Distributed Source Control

Centralized Source Control

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

- There is a single central copy of your project and programmers commit their changes to this central copy
- Common centralized version control systems are TFVC, CVS, Subversion (or SVN) and Perforce

Distributed Source Control

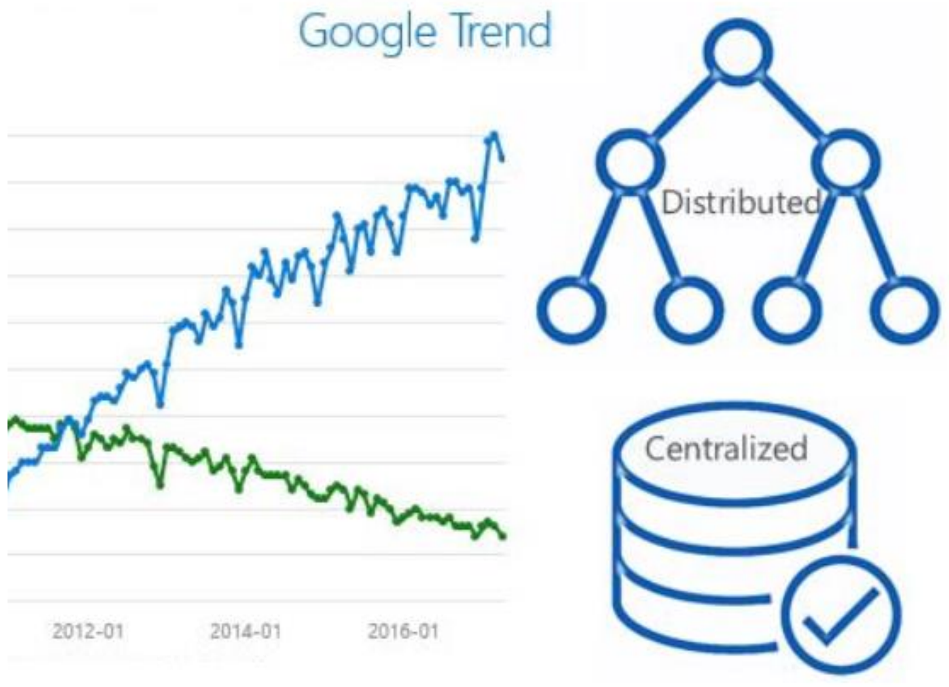


Strengths	Best Used for
• Cross platform support • An open source friendly code review model via pull requests • Complete offline support • Portable history • An enthusiastic growing user based	• Small and modular codebases • Evolving through open source • Highly distributed teams • Teams working across 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.

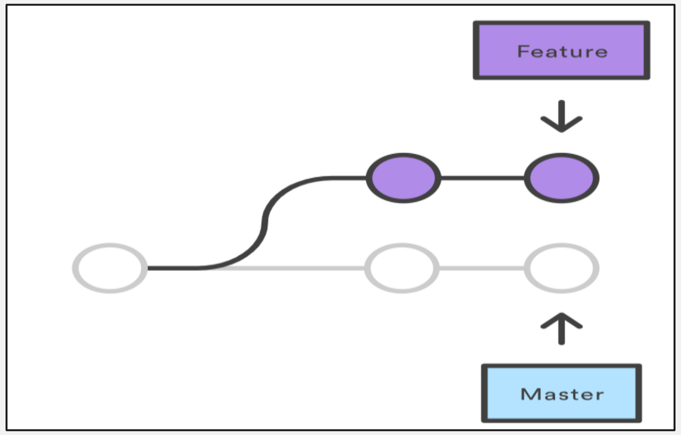
Git and TFVC

- Git
  - Distributed source control system
  - Each developer has a copy of the source repository on their dev machine
- 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

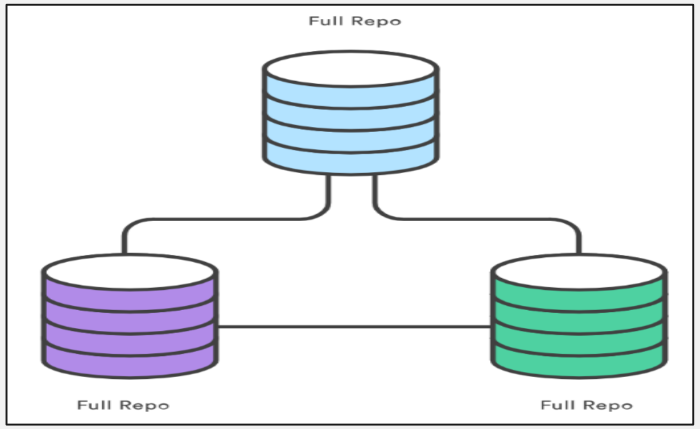


Why Git?

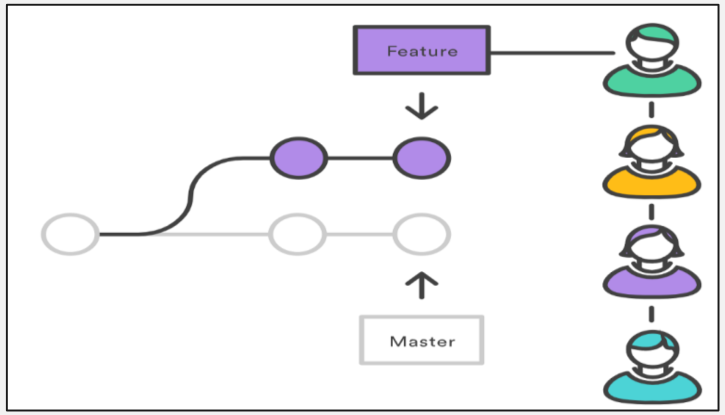
Feature branches



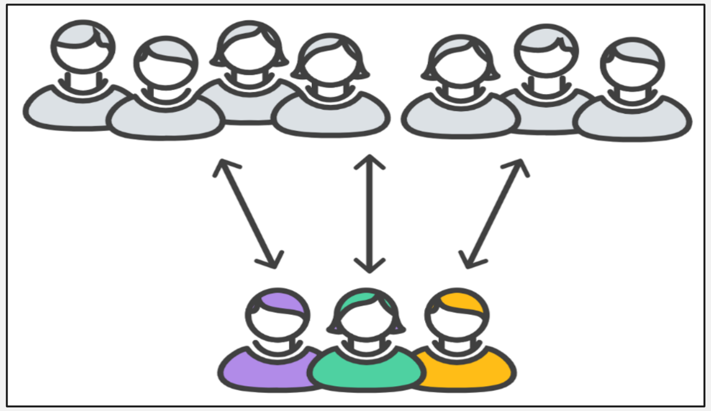
Distributed development



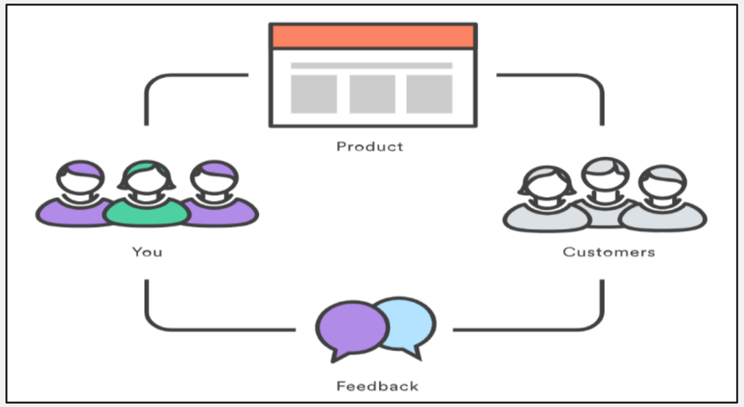
Pull requests



Community



Release cycles



Common Objections to Using Git

- Overwriting History – TFVC does not allow this, but with both you still overwrite the code
- Large Files - Git works best with repos that are small and that do not contain large files (or binaries); consider using Git LFS
- Large Repos – This is no longer a blocker
- Git? GitHub? – There is confusion about the difference
- Learning Curve – Some training and instruction will be needed

- Discussion: What objections do you have?

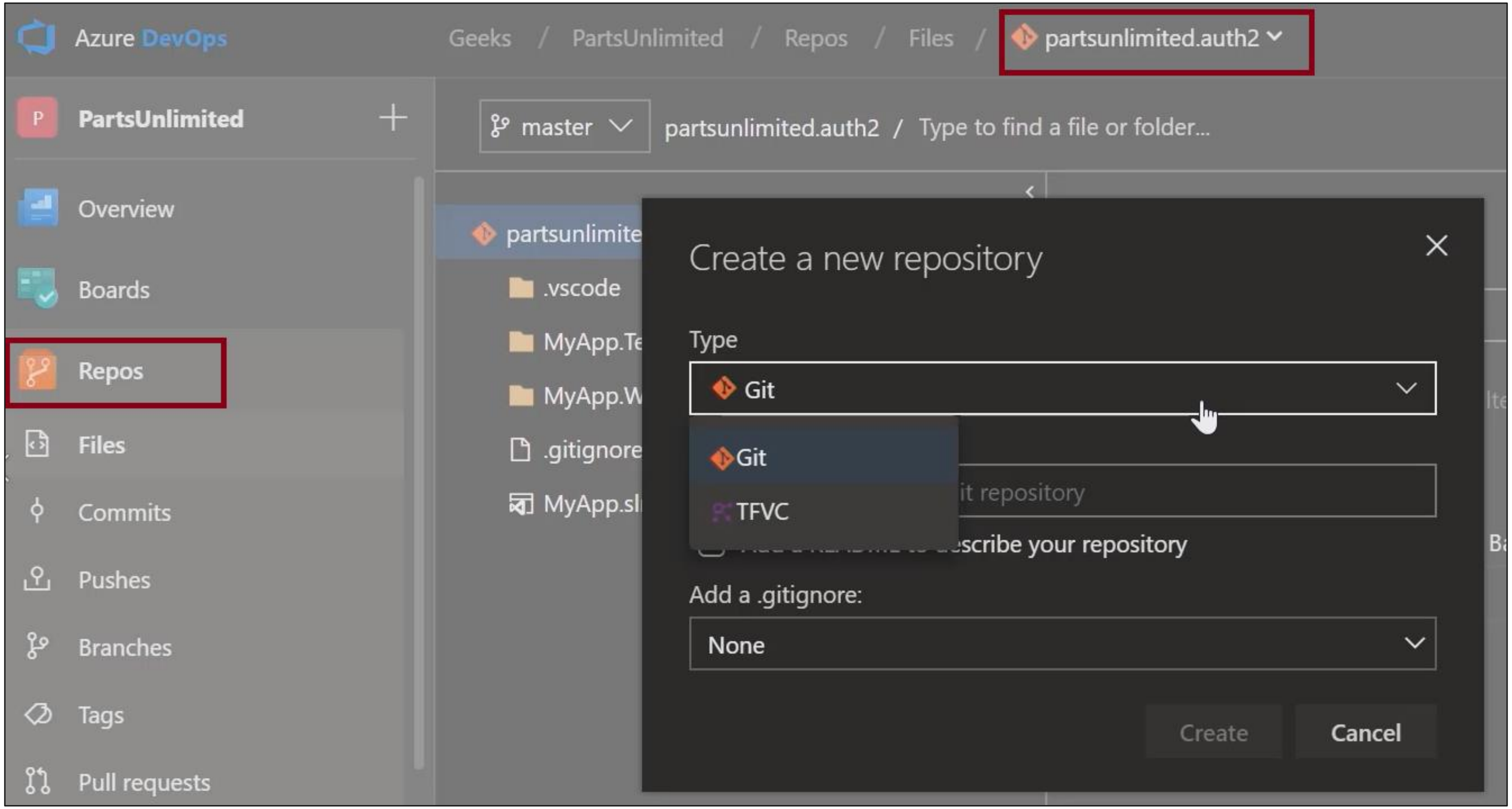
Working with Git Locally

1. Create a MVC web app and unit test project
2. Initialize a git repo
3. Create a new branch
4. Modify the code
5. Merge the changes
6. Review the history and rollback information

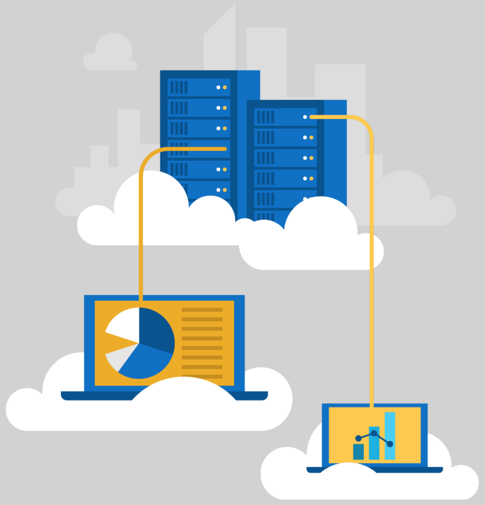
✓ All the above, are completed locally with git command line and seamlessly integrate with Visual Studio code

Lesson 04: Introduction to Azure Repos

Introduction to Azure Repositories

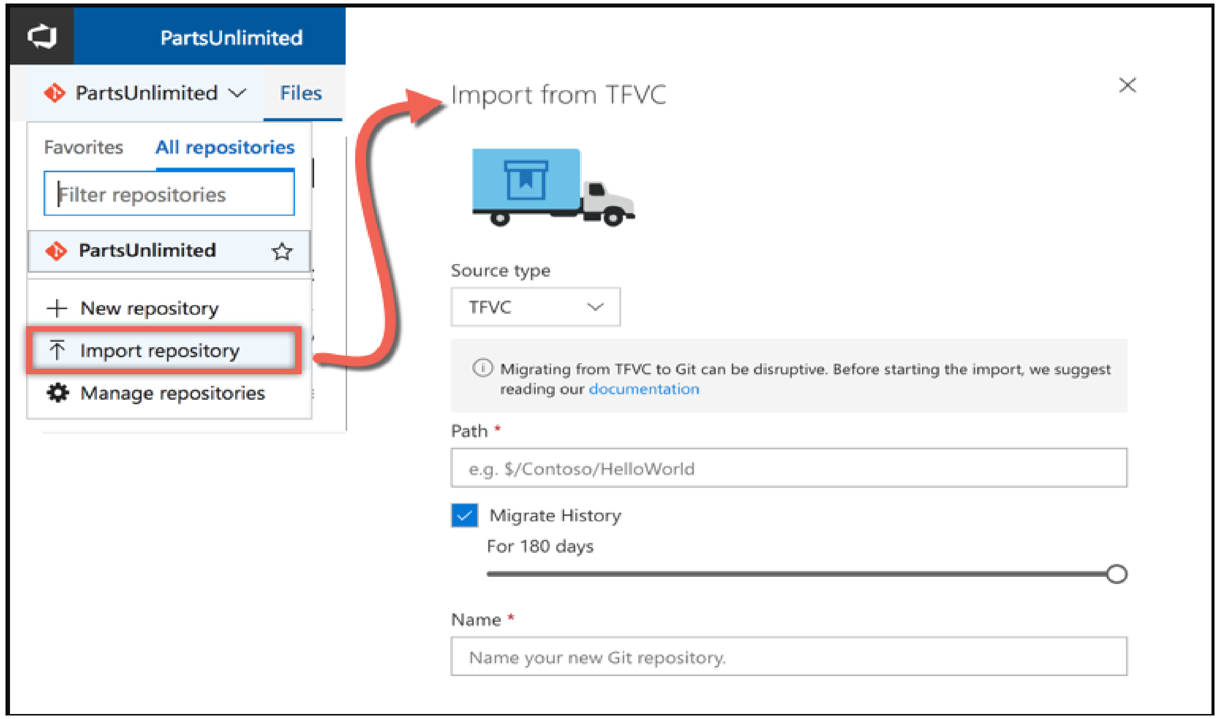


Lesson 05: Migrating from TFVC to Git



Migrating from TFVC to Git

- Single branch import



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.

- Full synchronization (Git-tfs)

Video: 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

Lab: Version Controlling with Git in Azure Repos

- In this lab, Version Controlling with Git in Azure Repos, you will establish and work with a local Git repository.
- You will learn how to:
  - Exercise 1: Cloning an existing repository
  - Exercise 2: Save work with commits
  - Exercise 3: Review history
  - Exercise 4: Manage branches from Visual Studio
  - Exercise 5: Managing branches from Azure DevOps

✓ Note that you must have already completed the prerequisite labs in the Welcome section.

Lesson 07: Module 1 Review Questions

Module 1: Review Questions

1. What is source control and what are the benefits of source control?
2. What are some best practices for source control?
3. What are the two basic types of source control and how do they work? When are the benefits and usage cases for each?