

AZ-400.00

Learning Path 01:

Get started on a DevOps transformation journey



Agenda



- Module 01: Introduction to DevOps.
- Module 02: Choose the right project.
- Module 03: Describe team structures.
- Module 04: Choose the DevOps tools.
- Module 05: Plan Agile with GitHub Projects and Azure Boards.
- Module 06: Introduction to source control.
- Module 07: Describe types of source control systems.
- Module 08: Work with Azure Repos and GitHub.
- Labs & Learning Path review and takeaways.

Learning Path overview



Learning objectives

After completing this Learning Path, students will be able to:

- 1 Plan for the transformation with shared goals and timelines
- 2 Select a project and identify project metrics and Key Performance Indicators (KPI's)
- 3 Create a team and agile organizational structure
- 4 Design a tool integration strategy
- 5 Design a license management strategy (e.g., Azure DevOps users)
- 6 Design a strategy for end-to-end traceability from work items to working software
- 7 Design an authentication and access strategy
- 8 Design a strategy for integrating on-premises and cloud resources

Learning objectives (continued)

After completing this Learning Path, students will be able to:

1 Describe the benefits of using Source Control

2 Describe Azure Repos and GitHub

3 Migrate from TFVC to Git

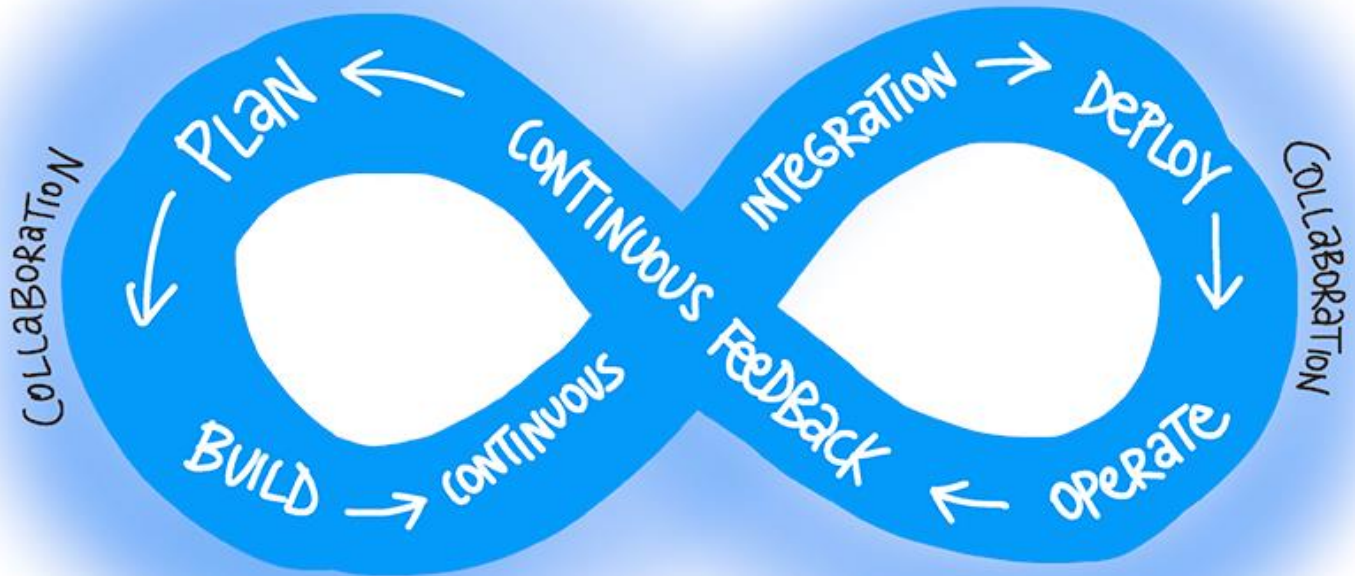
Module 01: Introduction to DevOps



What is DevOps?

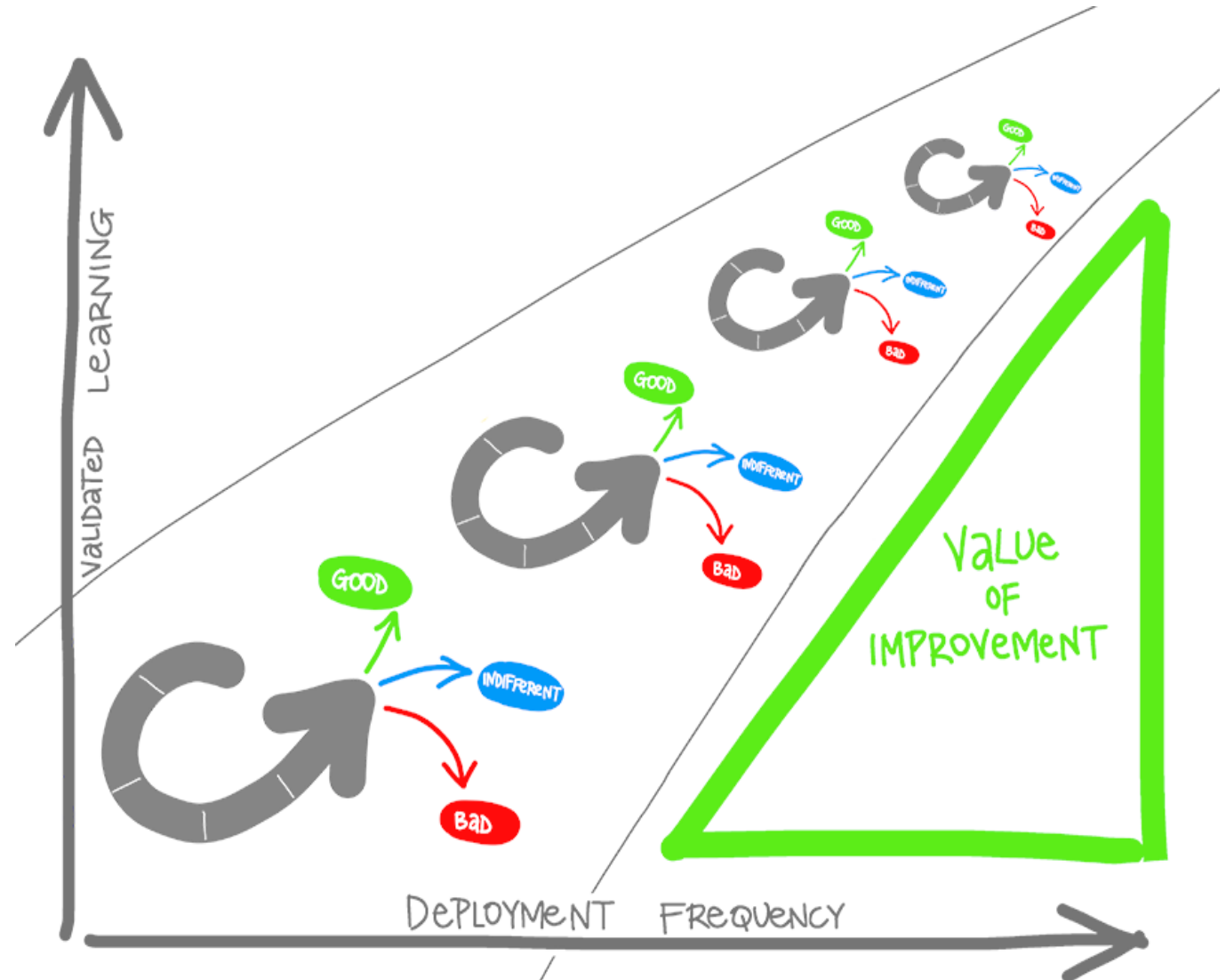
"DevOps is the union of people, process, and products to enable continuous delivery of value to end users."

— Donovan Brown,
[What is DevOps?](#)



What is DevOps? (continued)

- Understand your cycle time
- Become data-informed
- Strive for validated learning
- Shorten your cycle time
- Optimize validated learning



Explore the DevOps journey

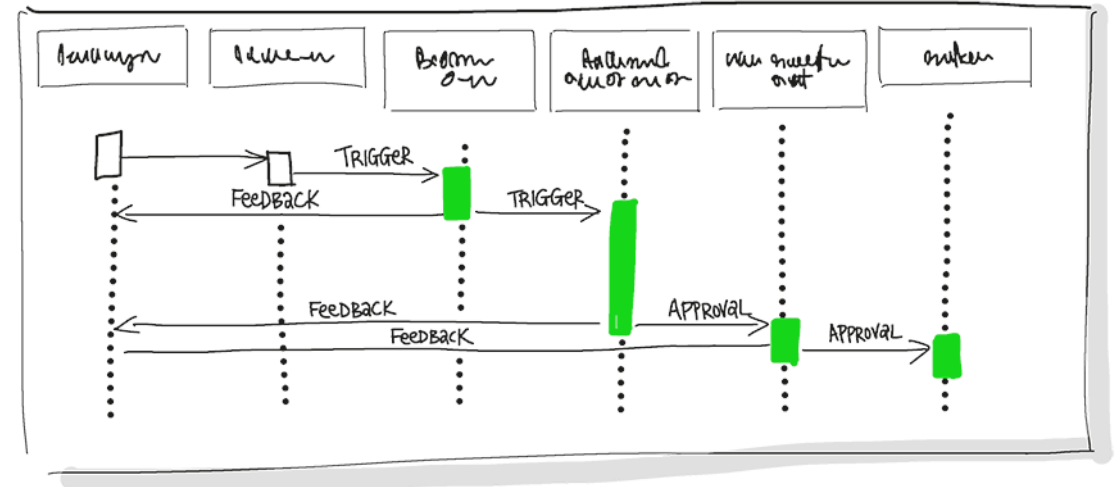
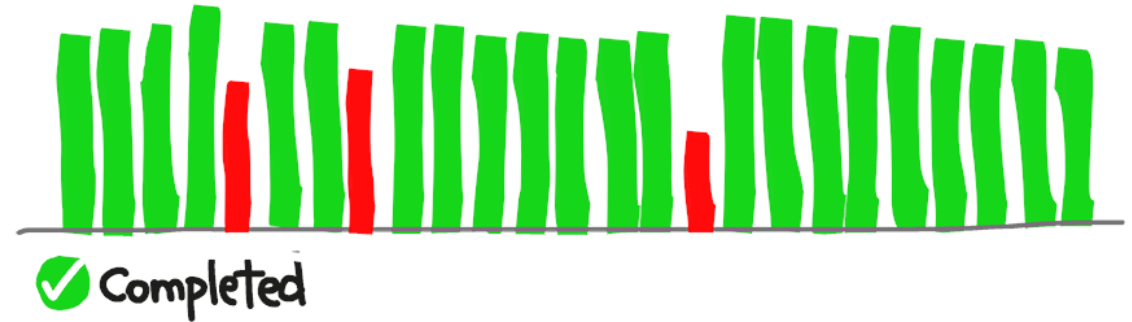
Continuous Integration

- Continuous Integration drives the ongoing merging and testing of code, which leads to finding defects early.

Continuous Delivery

- Continuous Delivery of software solutions to production and testing environments helps organizations quickly fix bugs and respond to ever-changing business requirements.

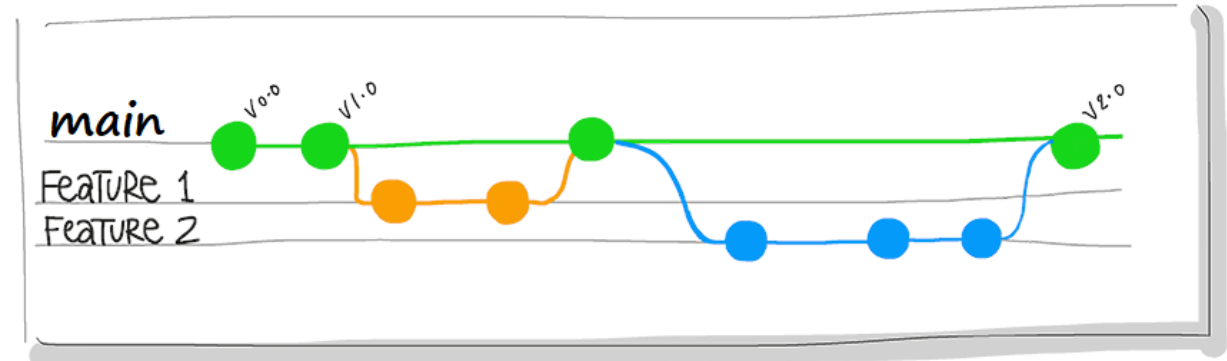
BUILD Succeeded



Explore the DevOps journey (continued)

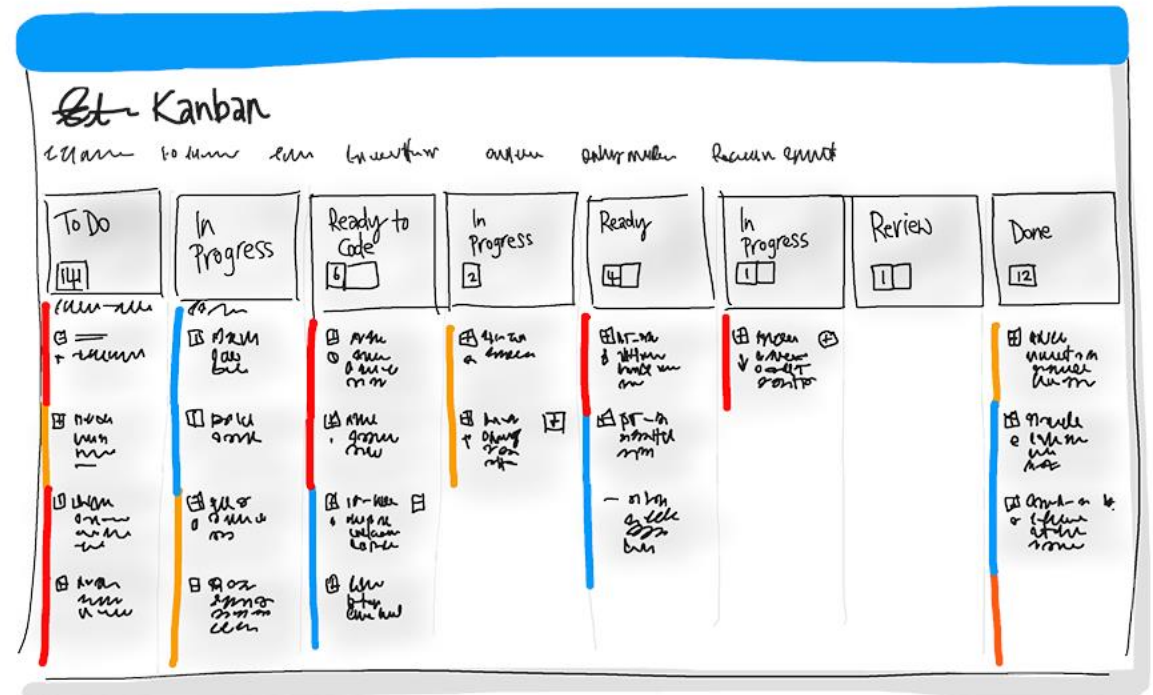
Version Control

- Version Control, usually with a Git-based Repository, enables teams located anywhere in the world to communicate effectively during daily development activities.



Agile/lean

- Plan and isolate work into sprints.
- Manage team capacity and help teams quickly adapt to changing business needs.
- A DevOps Definition of Done is working software collecting telemetry against the intended business goals.



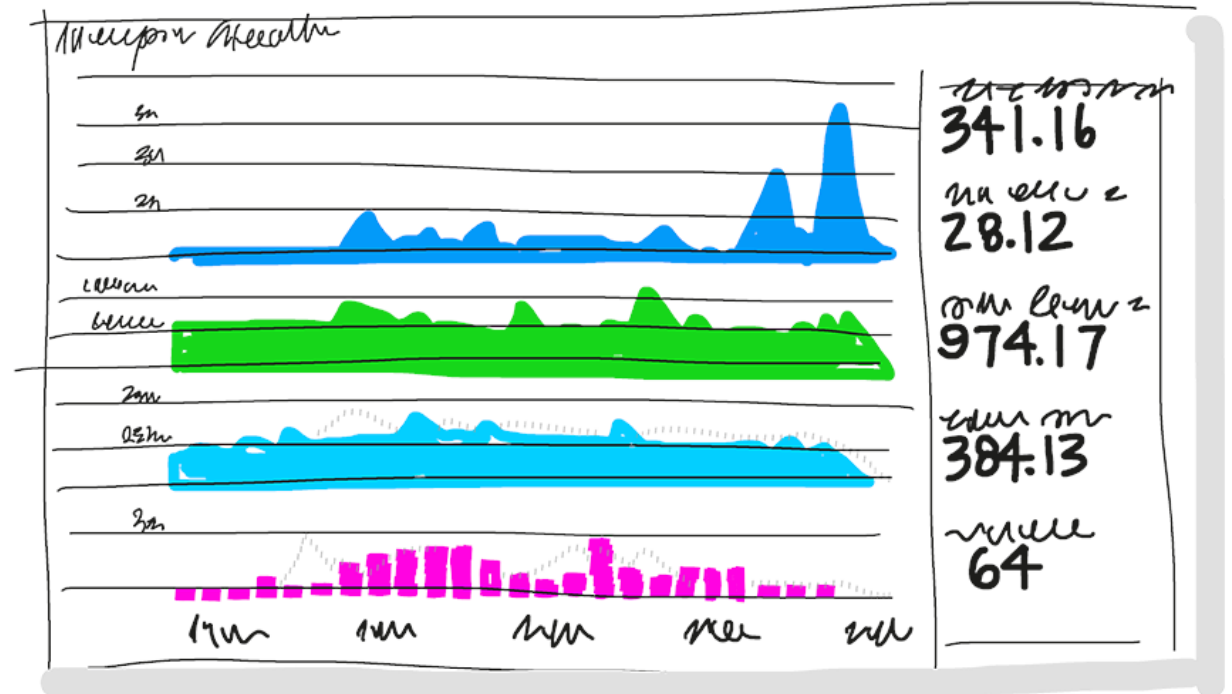
Explore the DevOps journey (continued)

Monitoring and logging

- Monitoring and Logging of running applications.

Cloud

- Public and Hybrid Clouds have made the impossible easy.



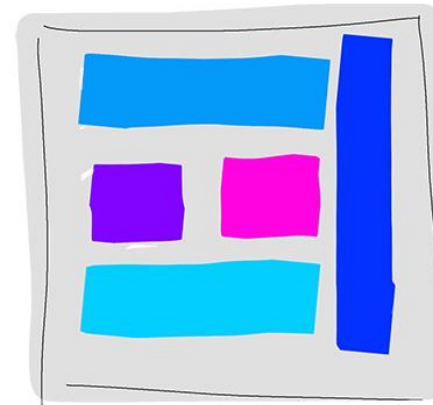
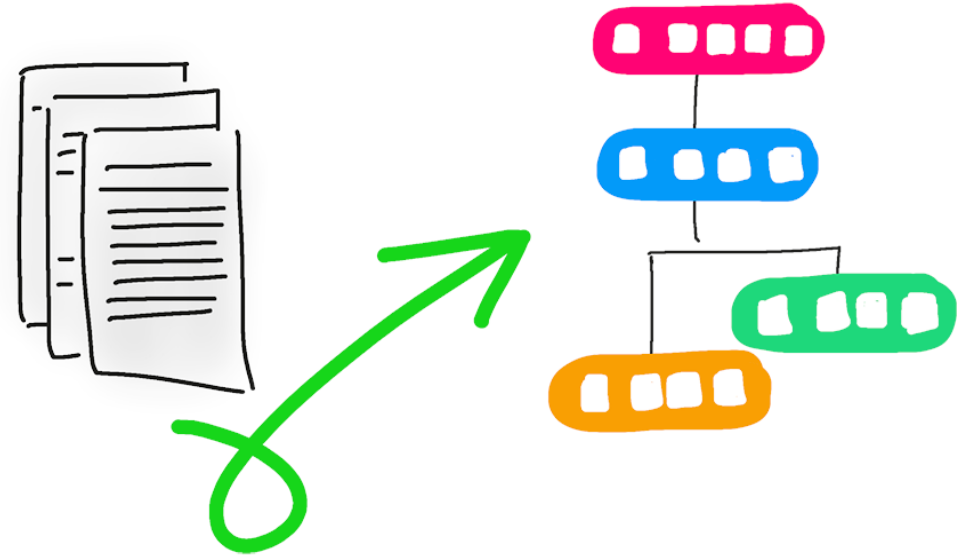
Explore the DevOps journey (continued)

Infrastructure as Code (IaC)

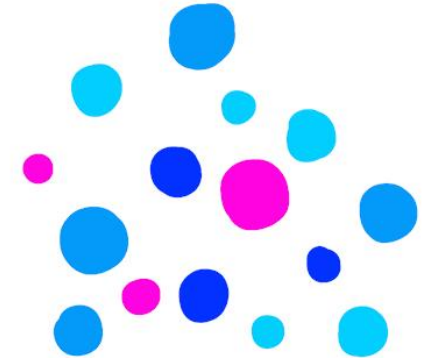
- Enables the automation and validation of the creation and teardown of environments to help with delivering secure and stable application hosting platforms.

Microservices

- Isolate business use cases into small reusable services that communicate via interface contracts.



MONOLITHIC/LAYERED



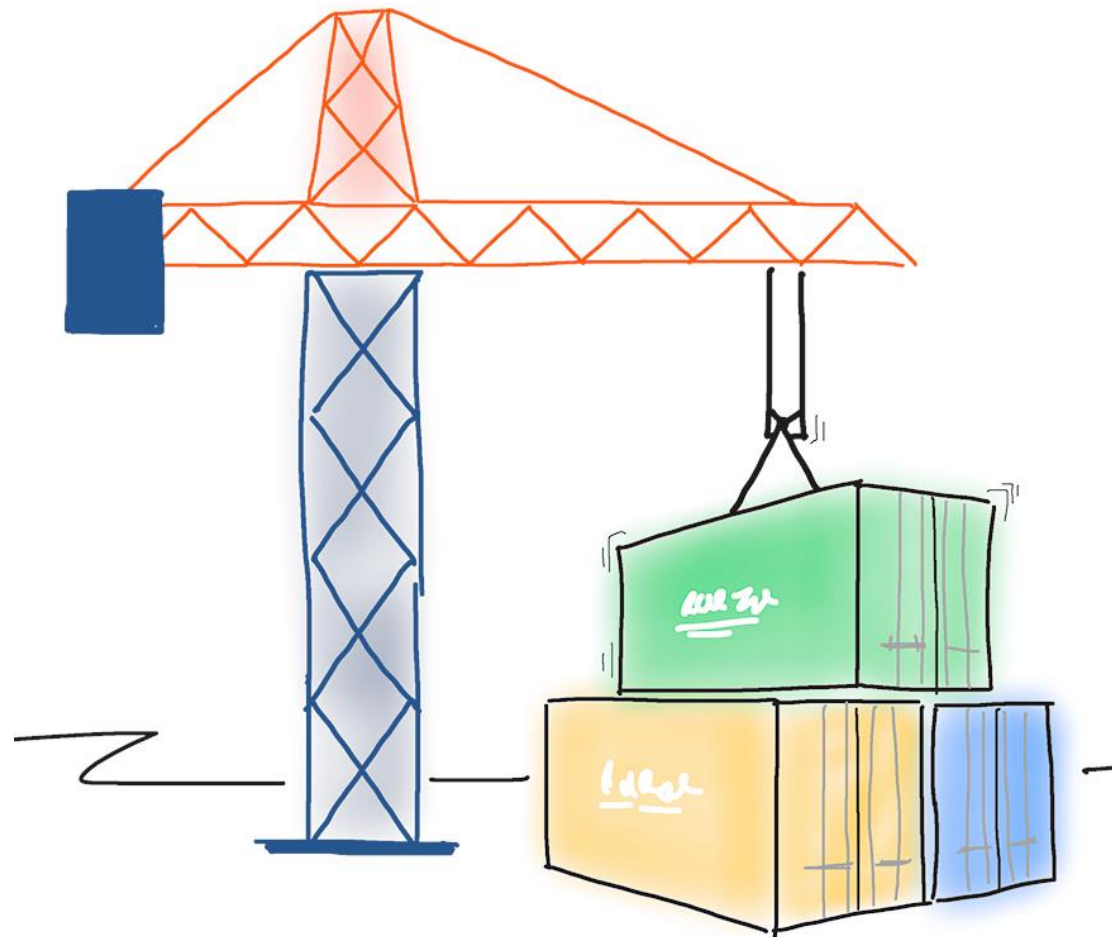
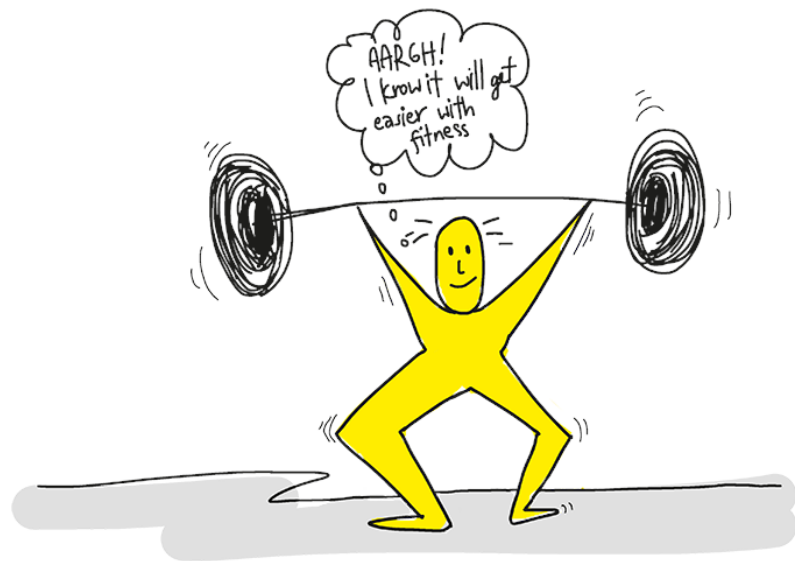
MICROSERVICES

Explore the DevOps journey (continued)

Containers

- Containers are the next evolution in virtualization.

DevOps may hurt at first



Explore agile development practices

1

Waterfall approach:

- Define, analyze, build and test, and deliver
 - Hard to accurately define requirements, which can change over time, including during development
 - Requires change requests and additional cost after delivery
-

2

Agile approach:

- Emphasizes constantly adaptive planning, and early delivery with continual improvement
- Development methods are based on releases and iterations
- At the end of each iteration, should have tested working code
- Is focused on shorter-term outcomes

Explore principles of agile development

- 1** Satisfy the customer through early and continuous delivery of valuable software
- 2** Welcome changing requirements
- 3** Deliver working software frequently
- 4** Work together throughout the project
- 5** Build projects around motivated individuals
- 6** Use face-to-face conversation
- 7** Measure progress through working software
- 8** Agile processes promote sustainable development
- 9** Continuous attention to technical excellence and good design
- 10** Simplicity – The art of maximizing the amount of work not done
- 11** Use self-organizing teams
- 12** Reflect on how to become more effective

Source: <https://www.agilealliance.org/agile101/12-principles-behind-the-agile-manifesto/>

Identify transformation teams

1

There are several challenges when creating teams:

- Availability of staff
 - Disruption of current procedures and processes
-

2

To overcome the challenges, create a team that is:

- Focused on the transformation
- Well respected in their subject areas
- Internal and external to the business



A transformation project can conflict with ongoing business needs

Explore shared goals and define timelines

1 Projects must have a clearly-defined set of measurable outcomes, like:

- Reduce the time spent on fixing bugs by 60%
- Reduce the time spent on unplanned work by 70%
- Reduce the out-of-hours work required by staff to no more than 10% of total working time
- Remove all direct patching of production systems



One of the key aims of DevOps is to provide greater customer value, so outcomes should have a customer value focus

Explore shared goals and define timelines (continued)

- 1** Measurable goals should have timelines that are challenging yet achievable
- 2** Timelines should be a constant series of short-term goals – each clear and measurable
- 3** Shorter timelines have advantages:
 - Easier to change plans or priorities when necessary
 - Reduced delay between doing the work and getting feedback
 - Easier to keep organizational support when positive outcomes are apparent

Module 02: Choose the right project



Explore greenfield and brownfield projects

- 1** Greenfield software projects develop in a totally new environment.
 - 2** Brownfield software projects develop in the immediate presence of existing software applications/systems.
-

Decide when to use greenfield and brownfield projects

1

Greenfield projects:

- Appears to be an easier starting point
 - A blank slate offers the chance to implement everything the way you want.
-

2

Brownfield projects:

- Comes with the baggage of existing code bases, existing teams and often a great amount of technical debt
- Spending time maintaining existing Brownfield applications, limits the ability to work on new code



There is a common misconception that DevOps suits greenfield projects better than brownfield projects, but this is not the case.

Decide when to use systems of record versus systems of engagement

1

Systems of record:

- Emphasize accuracy and security
 - Provide the truth about data elements
 - Historically evolve slowly and carefully
-

2

Systems of engagement:

- Are more exploratory
- Use experimentation to solve new problems
- Are modified regularly
- Prioritize making changes quickly over ensuring that the changes are correct



Both types of systems are important

Identify groups to minimize initial resistance

1

Different types of staff members:

- **Canaries** voluntarily test bleeding edge features
 - **Early adopters** voluntarily preview releases
 - **Users** consume the products after canaries and early adopters
-

2

Ideal Target Improvements:

- Can be used to gain early wins
- Are small enough to be achievable in a reasonable time-frame
- Have benefits that are significant enough to be obvious to the organization

Identify project metrics and key performance indicators (KPIs)

1

Faster outcomes – Deployment frequency, deployment speed, deployment size, and lead time

2

Efficiency – Server to admin ratio, staff member to customers ratio, application usage, and application performance

3

Quality and security – Deployment failure rates, application failure rates, mean time to recover, bug report rates, test pass rates, defect escape rate, availability, service level agreement (SLA) achievement, and mean time to detection

4

Culture – Employee morale and retention rates



Goals must be specific, measurable, and time-bound

Module 03: Describe team structures

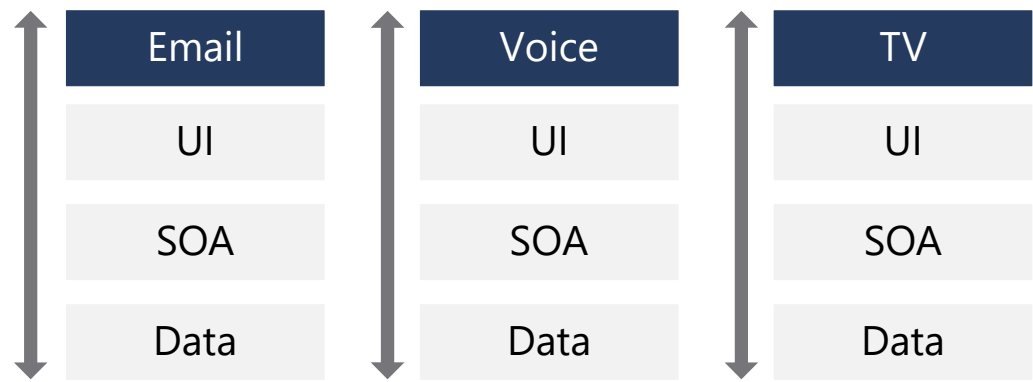


Define organization structure for agile practices

Horizontal team structures divide teams according to the software architecture.



Vertical teams span the architecture and are aligned with product outcomes, and scaling can occur by adding teams.



Vertical teams have been shown to provide stronger outcomes in Agile projects

Explore ideal DevOps team members

- 1
 - Think there is a need to change and have shown an ability to innovate
 - Are well-respected and have broad knowledge of the organization and how it operates
 - Ideally, already believe that DevOps practices are what is needed
-

- 2

Many teams hire external agile coaches or mentors

- 3

Agile coaches have teaching and mentoring skills

- 4

Agile coaches tend to be both trainers and consultants

- 5

Some coaches are technical experts

- 6

Some coaches are focused on agile processes



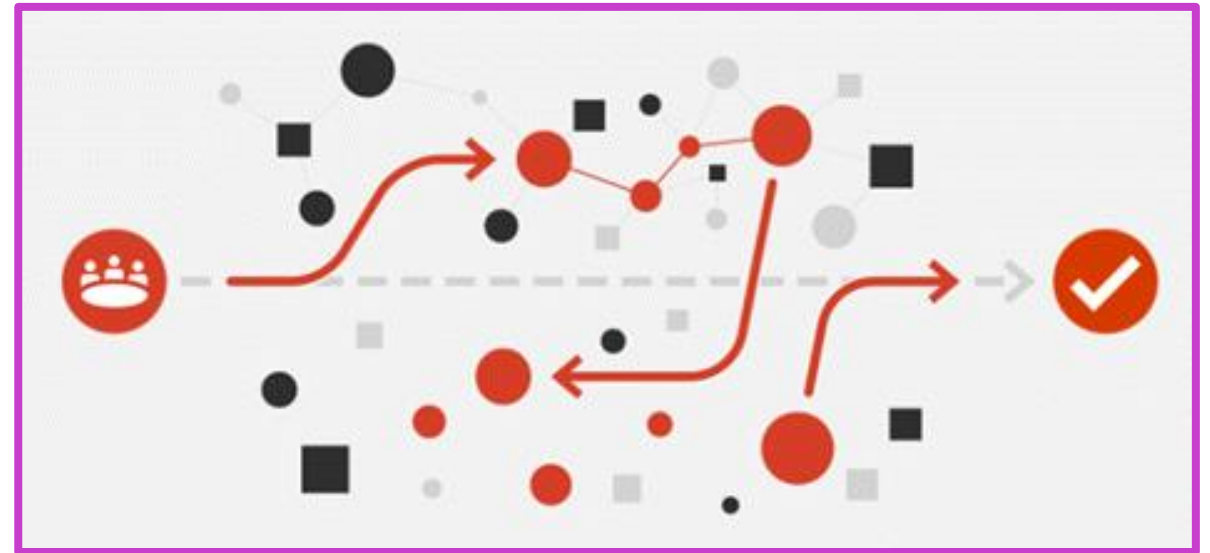
Team members must learn as they work, and acquire skills from each other

Enable in-team and cross-team collaboration

Cultural changes – More open workspaces, meeting etiquette, outsourcing, better communication

Cross-functional teams – Collaboration with others, diversity of opinion, rewarding collective behavior

Collaboration tooling – Slack, Teams, Asana, Glip, JIRA



Select tools and processes for agile practices

Tools can often enhance the outcomes achieved

Physical tools such as white boards, index cards, sticky notes

Project Management tools such as Kanban boards for planning, monitoring, and visualization

Screen recording tools for recording bugs, building walk-throughs, and demonstrations



Module 04: Choose the DevOps tools



What is Azure DevOps?

- 1 Azure Boards:** Agile planning, work item tracking, visualization and reporting tool
- 2 Azure Pipelines:** A language, platform and cloud agnostic CI/CD platform with support for containers or Kubernetes
- 3 Azure Repos:** Provides cloud-hosted private repos
- 4 Azure Artifacts:** Provides integrated package management with support for Maven, npm, Python and NuGet package feeds from public or private sources
- 5 Azure Test Plans:** Provides an integrated planned and exploratory testing solution

What is GitHub?

- 1** **Codespaces:** Provide cloud-hosted collaborative development environments
- 2** **Repos:** Provide cloud-hosted and on-premises git repos for both public and private projects
- 3** **Actions:** Create automation workflows with environment variables and customized scripts
- 4** **Packages:** Ease integration with numerous existing packages and open-source repositories
- 5** **Security:** Review code and identity vulnerabilities early in the development cycle

Explore an authorization and access strategy

1

Azure DevOps Services uses either a Microsoft account or Microsoft Entra ID, to protect and secure your data

2

For non-Microsoft tools like Git, NuGet, or Xcode you can use personal access tokens

3

Azure DevOps is pre-configured with default security groups and permissions

4

You can also configure app access policies and conditional access policies

Migrate or integrate existing work management tools

Both Azure DevOps and GitHub can be integrated with a wide variety of existing work management tools:

- [Trello integration tooling](#) is a free, flexible, and visual way to organize anything with anyone.
- [Solidify](#) offers a tool for Jira to Azure DevOps migration.
- Third party organizations offer commercial tooling to assist with migrating other work management tools like Aha, BugZilla, ClearQuest.

Migrate or integrate existing test management tools

1 Azure Test Plans are used to track sprints and milestones. There is a Test & Feedback extension available in the Visual Studio Marketplace.

2 Other tools:

- [Apache JMeter](#) is open-source software written in Java and designed to load test functional behavior and measure performance.
- [Pester](#) is a tool that can be used to automate the testing of PowerShell code.
- [SoapUI](#) provides another testing framework for SOAP and REST testing.



If you are using Microsoft Test Manager, you should plan to migrate to Azure Test Plans

Design a license management strategy

1

Azure DevOps can be licensed for individual services or for users. It offers free and paid tiers:

- <https://azure.microsoft.com/en-us/pricing/details/devops/azure-devops-services/>
-

2

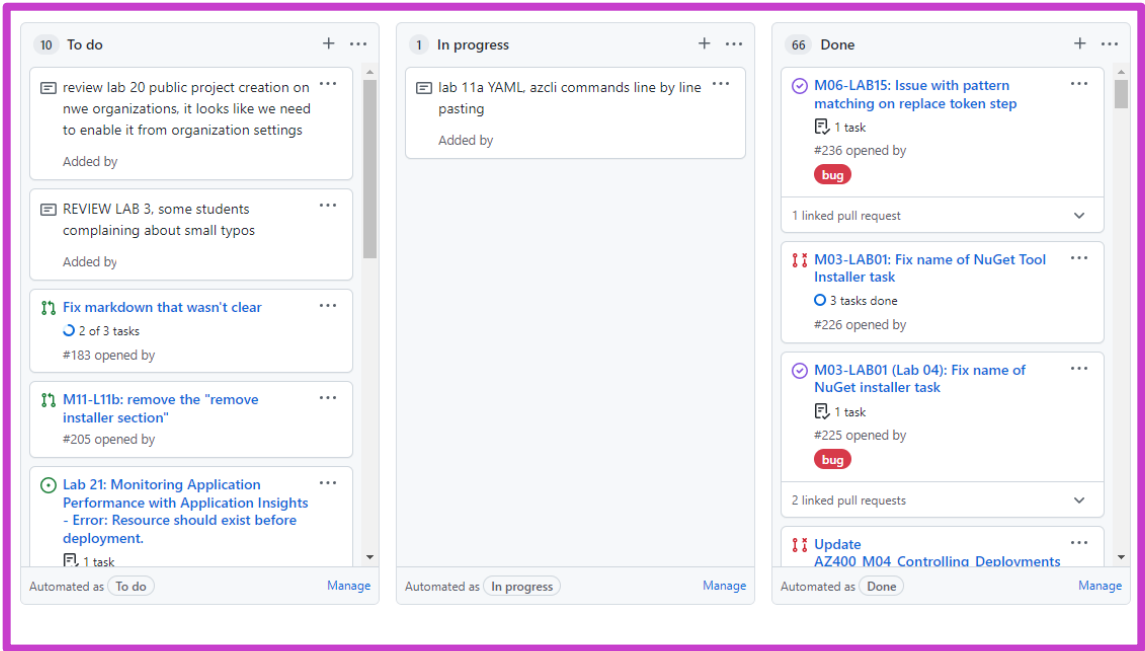
GitHub can be licensed for individuals, teams, and enterprises. It offers free and paid tiers:

- <https://github.com/pricing>

Module 05: Plan Agile with GitHub Projects and Azure Boards



Introduction to GitHub Projects and Project boards

























All work

Ready for review

Board

New view

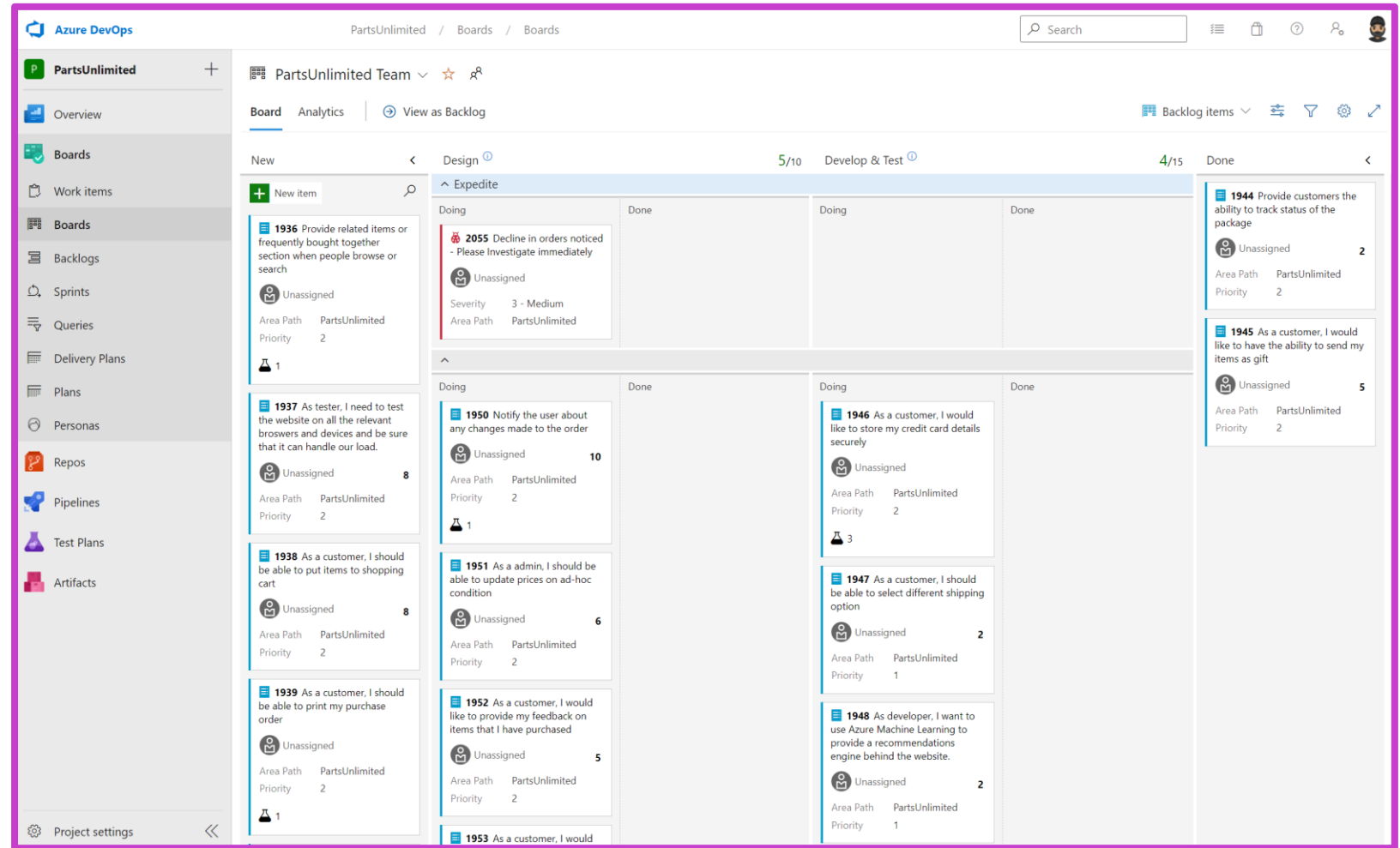
	Title	Assignees	Status
1	 Add Travis CI migration table	 octocat	Ready for review
2	 Using a package without installing package	 monalisa	Ready 
3	 Add functionality to hide images	 octocat	In Progress 
4	 Update contributing guide	 octocat	Ready for review
5	 Fix markdown tables in translated content	 monalisa	Ready 
6	 add copy button to examples	 octocat	In Progress 
7	 Validate Java Gradle wrapper		Todo 
8	 GPC creation on linux		Todo 
9	 Add domains for self-hosted runners		Todo 

GitHub Project Boards Project boards are made up of issues, pull requests, and notes categorized as cards that you can drag and drop into your chosen columns.

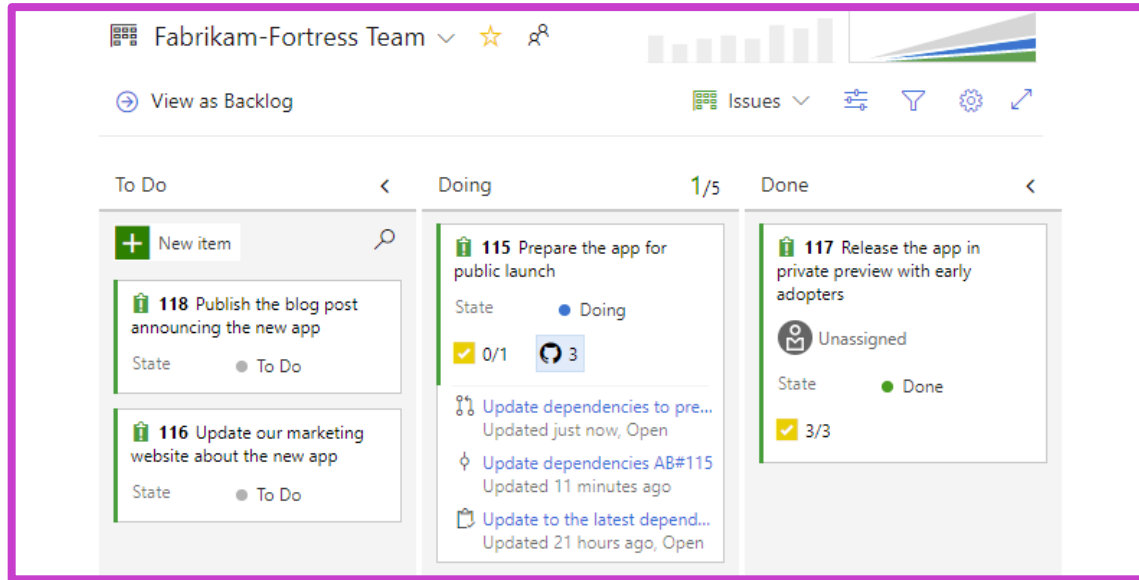
GitHub Projects are a new, customizable and flexible tool version of projects for planning and tracking work on GitHub.

Introduction to Azure Boards

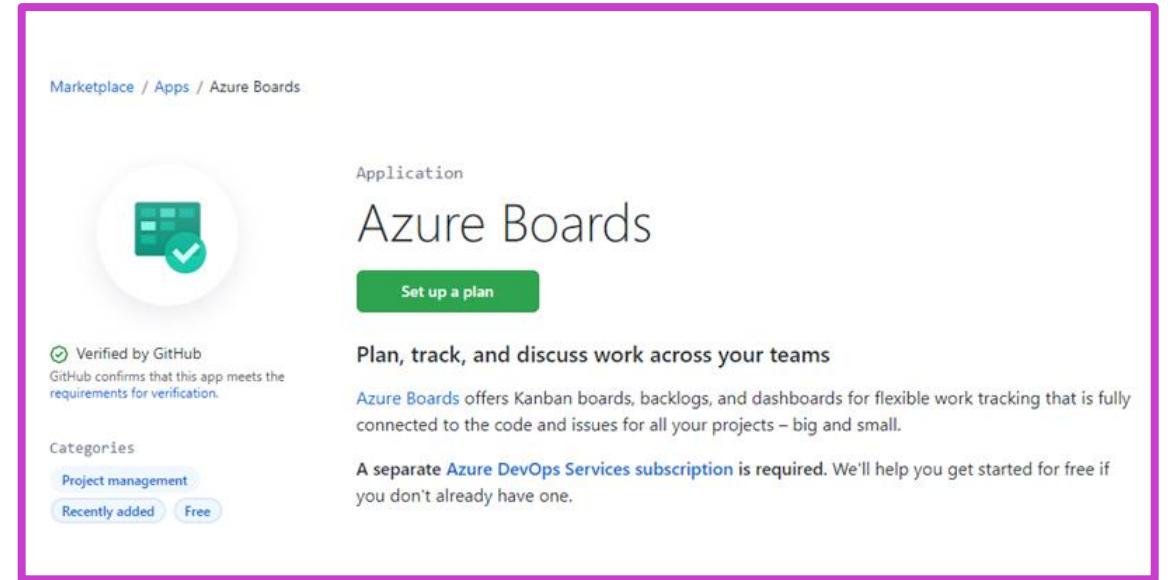
- Agile, Scrum, and Kanban processes by default.
- Track work, issues, and code defects associated with your project.



Link GitHub to Azure Boards



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

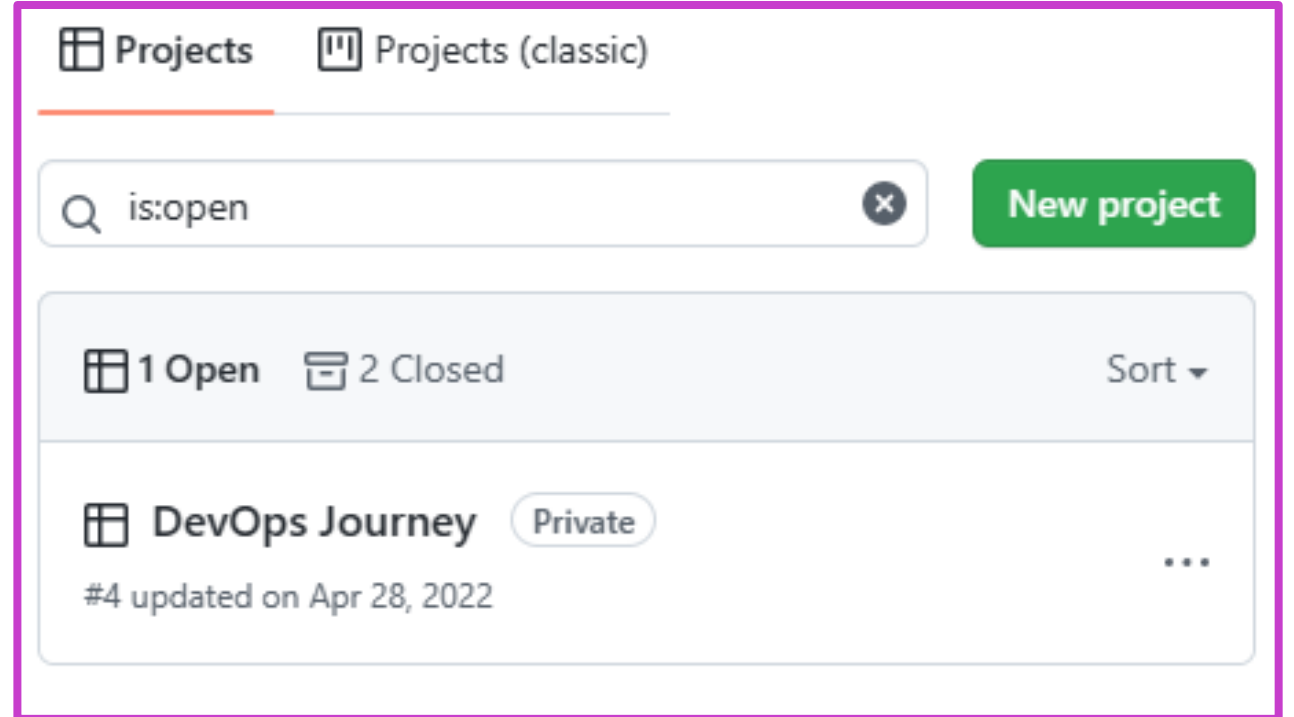


Use Azure Boards to plan and track your work, and GitHub as source control for software development.

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

Configure GitHub Projects

- You can use a project board template to create a project board with automation already configured.
- You can also copy a project board to reuse its customizations for similar projects.
- You can link up to twenty-five repositories to your organization or user-owned project board.
- Once you've created your project board, you can add issues, pull requests, and notes to it.



Manage work with GitHub Project boards

- Control project deliverables
- Set to any length of time
- Include breaks
- Release dates
- Iterations to planning upcoming work

The screenshot displays the GitHub Project board interface. At the top, there are two summary items: '3 Active' with a clock icon and '4 Completed' with a checklist icon. To the right is a button labeled '+ Add iteration' with a dropdown arrow. Below this is a list of four items, each with a status label in a pill, a title, a duration, and a date range. The first item is 'Current' (blue pill), 'Iteration 5', '2 weeks', 'Feb 23 - Mar 08'. The second is 'Planned' (grey pill), 'Iteration 6', '2 weeks', 'Mar 09 - Mar 22'. The third is 'Break' (grey pill), '1 week', 'Mar 23 - Mar 29'. The fourth is 'Planned' (grey pill), 'Iteration 7', '2 weeks', 'Mar 30 - Apr 12'. Each item has a trash icon on the right. A blue pill labeled 'Insert break' is positioned between the first and second items. At the bottom are two buttons: 'Save changes' (green) and 'Reset' (grey).
















Status	Iteration	Duration	Start Date	End Date	Action
Current	Iteration 5	2 weeks	Feb 23	Mar 08	Trash
Planned	Iteration 6	2 weeks	Mar 09	Mar 22	Trash
Break	1 week	Mar 23	Mar 29		Trash
Planned	Iteration 7	2 weeks	Mar 30	Apr 12	Trash

Customize Project views

Organize information by changing the layout, grouping, sorting, and filtering your work.

Use Command Palette:

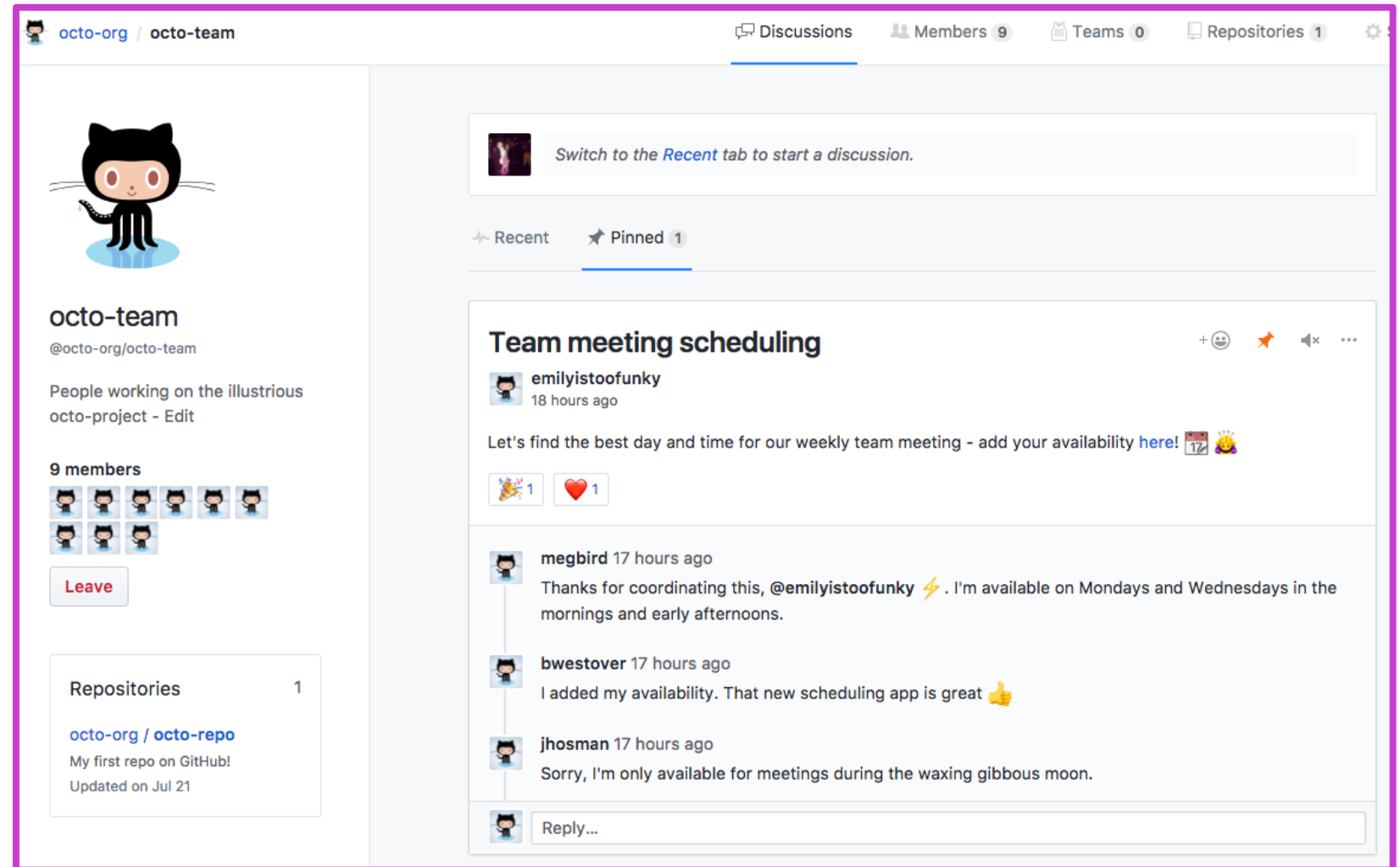
- Switch layout: Table.
- Show: Milestone.
- Sort by: Assignees, asc.
- Group by: Status.
- Column field by: Status.
- Filter by Status.
- Delete view.

All work ▾			Ready for review	Board	+ New view
Q	Title	Assignees			
1	 Add Travis CI migration table	 octocat			
2	 Using a package without installing package	 monalisa			
3	 Add functionality to hide images	 octocat			
4	 Update contributing guide	 octocat			
5	 Fix markdown tables in translated content	 monalisa			
6	 add copy button to examples	 octocat			
7	 Validate Java Gradle wrapper				
8	 GPC creation on linux				
9	 Add domains for self-hosted runners				

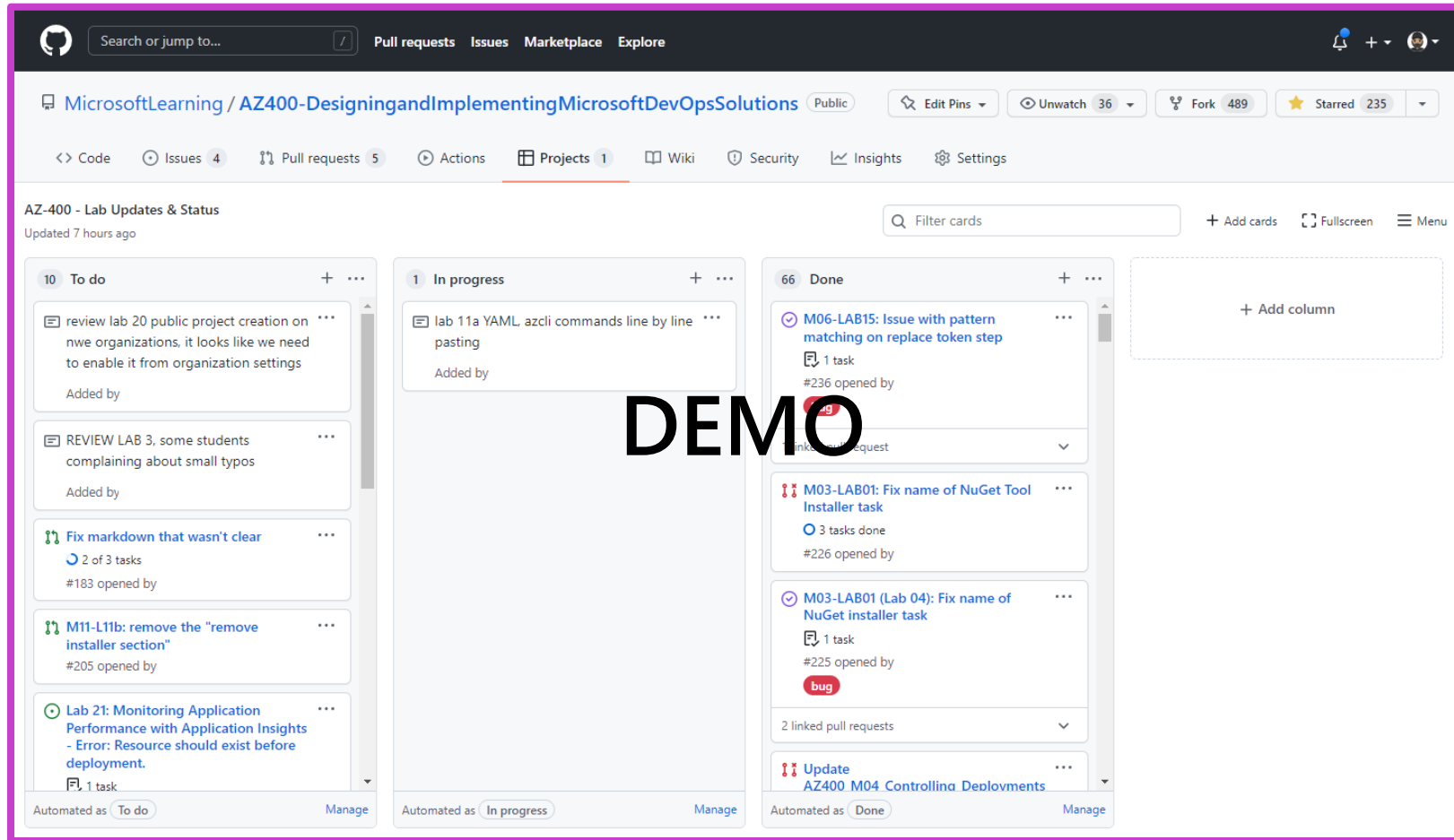
Collaborate using team discussions

GitHub discussions can help make your team plan together, update one another, or talk about any topic.

Across projects or repositories conversations (issues, pull requests, etc.).



Demonstration: GitHub Project Boards and Projects



Module 06: Introduction to source control



Introduction

- 1 When it was introduced, DevOps was a revolutionary way to release software quickly and efficiently while maintaining a high level of security.
- 2 Source control (version control) is a critical part of DevOps.

Explore DevOps foundational practices

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

	Defining practices and associated practices	Practices that contribute to success
Stage 0	<ul style="list-style-type: none">• 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	<ul style="list-style-type: none">• Application development teams use version control• Teams deploy on a standard set of operating systems	<ul style="list-style-type: none">• 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	<ul style="list-style-type: none">• Build on a standard set of technologies• Teams deploy on a single standard operating system	<ul style="list-style-type: none">• Deployment patterns for building applications and services are reused• Rearchitected applications based on business needs• Put system configurations in version control

Explore DevOps foundational practices (continued)

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

	Defining practices and associated practices	Practices that contribute to success
Stage 3	<ul style="list-style-type: none">• 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	<ul style="list-style-type: none">• 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	<ul style="list-style-type: none">• System configurations are automated• Provisioning is automated• Application configurations are in version control• Infrastructure teams use version control	<ul style="list-style-type: none">• Security policy configurations are automated• Resources made available via self-service
Stage 5	<ul style="list-style-type: none">• Incident responses are automated• Resources available via self-service• Rearchitected applications based on business needs• Security teams are involved in technology design and deployment	<ul style="list-style-type: none">• Security policy configurations are automated• Application developers deploy testing environments on their own• Success metrics for projects are visible• Provisioning is automated

What is source control?

1

A source control system allows developers to collaborate on code and track changes. Use version control to save your work and coordinate code changes across your team.

2

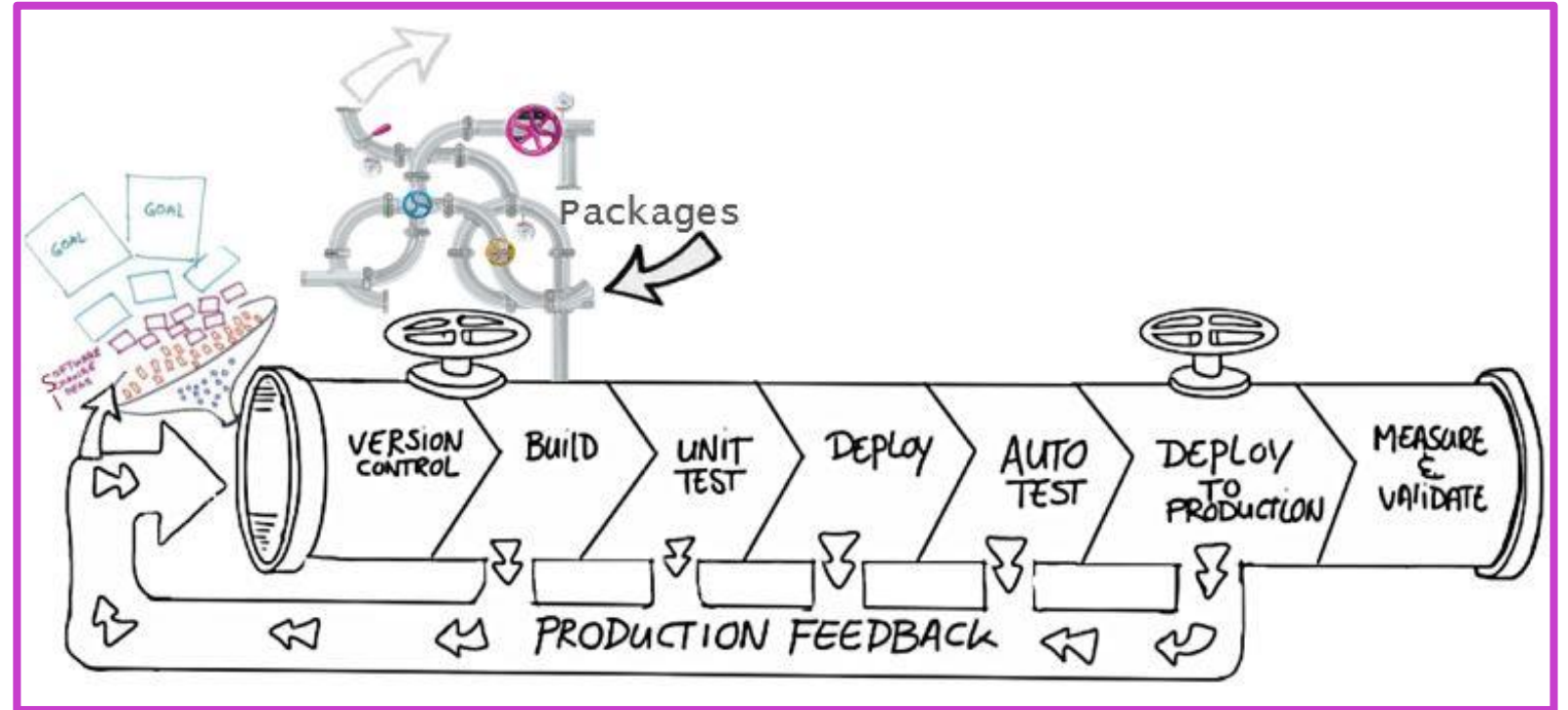
The version control system saves a snapshot of your files (history) so that you can review and even roll back to any version of your code with ease.

3

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

Explore benefits of source control

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



Explore best practices for source control

- 1 Make small changes

- 2 Don't commit personal files

- 3 Update often and right before pushing to avoid merge conflicts

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

- 5 Pay close attention to commit messages as these will tell you why a change was made

- 6 Link code changes to work items

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

Module 07: Describe types of source control systems



Understand centralized source control



Centralized

Strengths

- Easily scales for very large codebases
- Granular permission control
- Permits monitoring of usage
- Allows exclusive file locking

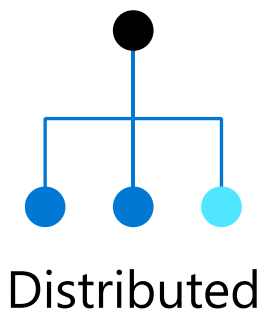
Best Used for

- 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.

Understand distributed source control



Strengths	Best Used for
<ul style="list-style-type: none">• Cross platform support• An open-source friendly code review model via pull requests• Complete offline support• Portable history• An enthusiastic growing user base	<ul style="list-style-type: none">• 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 or Git.

Explore Git and Team Foundation Version Control

Git:

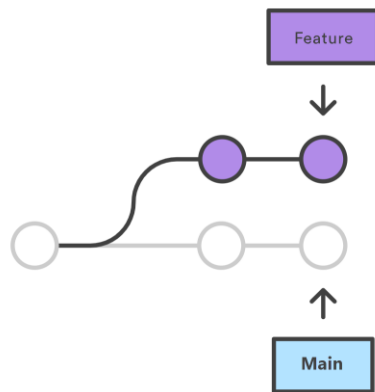
- Distributed Source Control system.
- Each developer has a copy of the source repository on their development system.

TFVC:

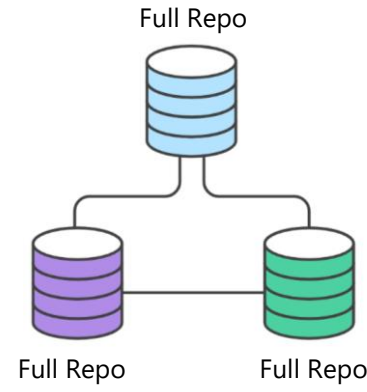
- Centralized Source Control system.
- Team members have only one version of each file on their dev machines.
- In the **server workspaces** model, before making changes, team members publicly check out files.
- In the **local workspaces** model, each team member takes a copy of the latest version of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

Examine and choose Git

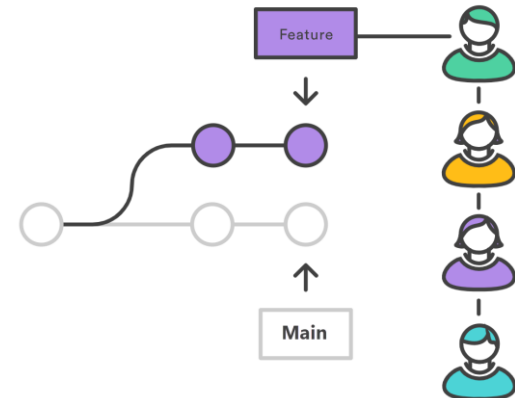
Feature branches



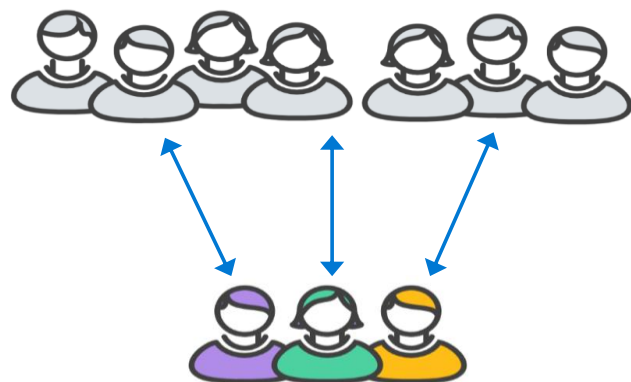
Distributed development



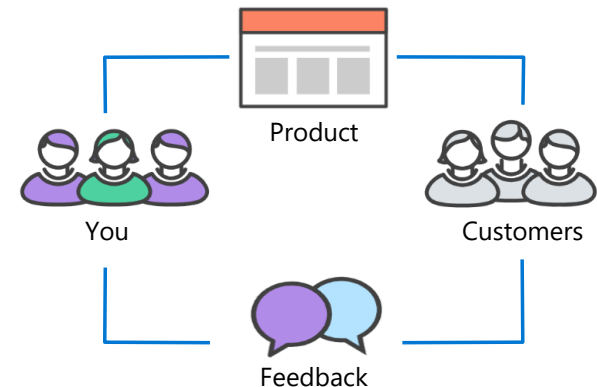
Pull requests



Community



Release cycles



Understand objections to using Git

- 1** **Overwriting History** – If used incorrectly, it can lead to conflicts.

- 2** **Large Files** – Git works best with repos that are small and that do not contain large files (or binaries); consider using Git LFS support.

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

- 4** **Discussion** – What objections do you have?

Describe working with Git locally

DEMO

Module 08: Work with Azure Repos and GitHub



Introduction to Azure Repos

1

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

Introduction to GitHub

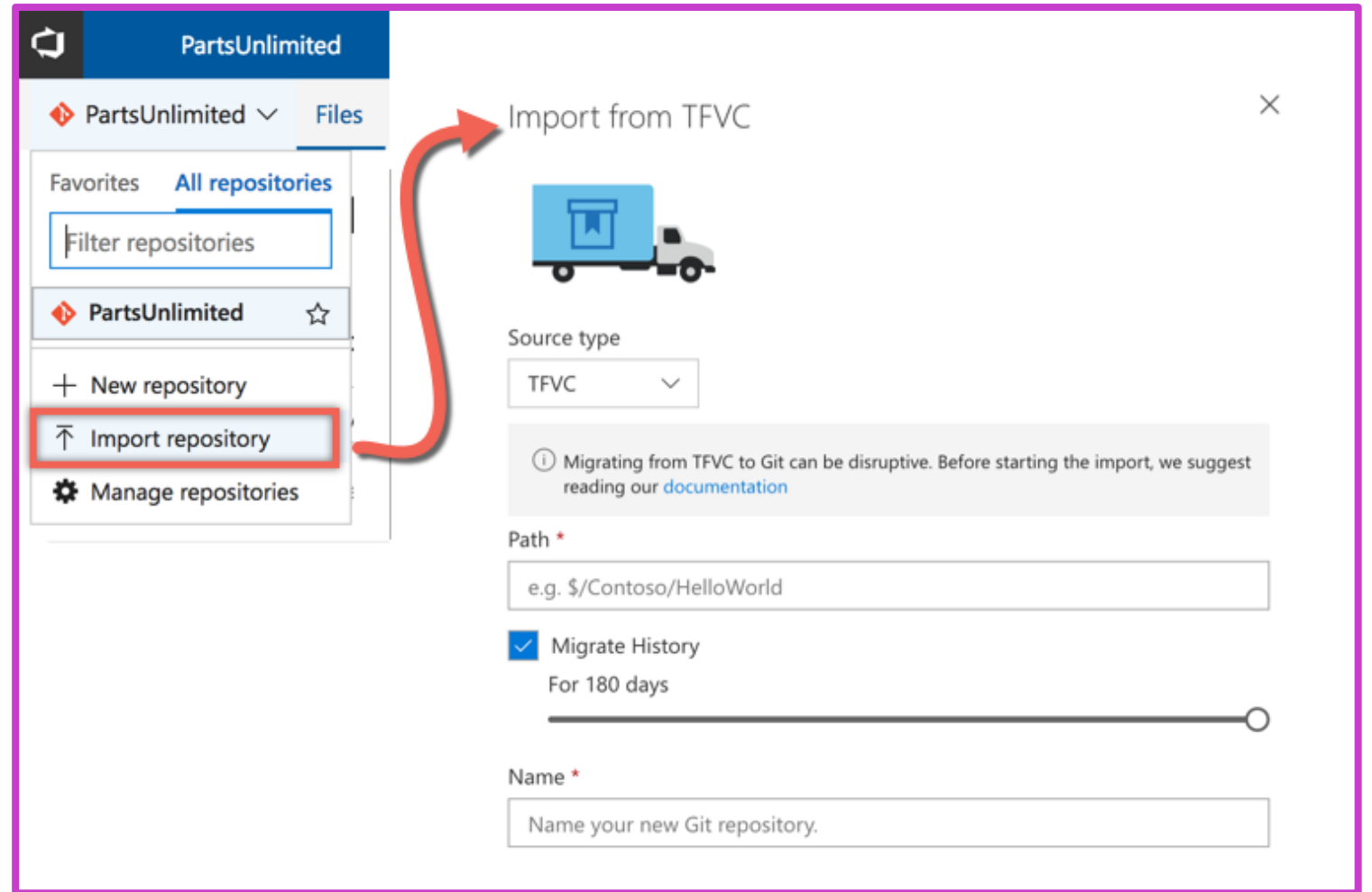
1 Largest open-source community

2 Features of GitHub:

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

Migrate from TFVC to Git

- Single branch import
- Full synchronization (git-tfs)



Migrate from TFVC to Git (continued)

1

Migrating the tip:

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

2

Migrating with history:

Tries to mimic the history in Git

3

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

Develop online with GitHub Codespaces

Cloud-based development environment hosted by GitHub

1

Avoids issues with old hardware/software

2

Highly portable

3

Protect against proliferation of intellectual property

4

Based on Visual Studio Code

5

Work from PCs, tablets, Chromebooks

6

Connect to Codespaces from Visual Studio Code

Labs



Lab: Agile planning and portfolio management with Azure Boards



Lab overview:

In this lab you will learn about the agile planning and portfolio management tools and processes provided by Azure Boards and how they can help you quickly plan, manage, and track work across your entire team.

Objectives:

- Manage teams, areas, and iterations
- Manage work items
- Manage sprints and capacity
- Customize Kanban boards
- Define dashboards
- Customize team process

Duration:



Learning Path review and takeaways



What did you learn?

1

Plan for the transformation with shared goals and timelines

2

Select a project and identify project metrics and Key Performance Indicators (KPI's)

3

Create a team and agile organizational structure

4

Design a tool integration strategy

5

Design a license management strategy (e.g., Azure DevOps and GitHub users)

6

Design a strategy for end-to-end traceability from work items to working software

7

Design an authentication and access strategy

8

Design a strategy for integrating on-premises and cloud resources

What did you learn? (continued)

9 Describe the benefits of using source control

10 Describe Azure Repos and GitHub

11 Migrate from TFVC to Git

Learning Path review questions

- 1 Which of the following would a system that manages inventory in a warehouse be considered?
- 2 An Agile tool that is used to manage and visualize work by showing tasks moving from left to right across columns representing stages. What is this tool commonly called?
- 3 In which of the following would you find large amounts of technical debt?
- 4 As a project metric, what is Lead Time measuring?
- 5 What is a cross-functional team?

Learning Path review questions (continued)

- 6 What are some of the benefits of source control?
- 7 What are the benefits of using distributed version control?
- 8 What are the benefits of using centralized version control?
- 9 What is source control?

