AZ-400: Get started on a DevOps transformation journey

Introduction to DevOps

Introduction

Completed

- 4 minutes

"DevOps is the union of people, process, and products to enable continuous delivery of value to our end users." - According to Donovan Brown in What is DevOps?

The DevOps learning paths will help you prepare for a DevOps journey. You'll learn the main characteristics of the DevOps process, tools, and people involved during the lifecycle. Also, it prepares you for the Microsoft DevOps Solution certification exam. You'll see other content to ensure you have a complete picture of DevOps. The module's content includes graphics, reference links, module review questions, and optional hands-on labs.

You'll learn the following:

- How to plan for DevOps.
- Use source control.
- Scale Git for an enterprise.
- Combine artifacts.
- Design a dependency management strategy.
- Manage secrets.
- Implement continuous integration.
- Implement a container-build strategy.
- Design a release strategy.
- Set up a release management workflow.
- Implement a deployment pattern.
- Optimize feedback mechanisms.

Plan before you act. This module will help you understand what DevOps is and how to plan for a DevOps transformation journey.

What is the DevOps transformation journey?

The DevOps transformation journey is a series of 9 learning paths. It familiarizes you with Azure DevOps and GitHub. Also, learn its many services, features, and integration with tools to support your DevOps process.

Why should I take the DevOps learning path?

People in these modules are interested in designing and implementing DevOps processes. Also, they're preparing for the AZ-400 - Designing and Implementing Microsoft DevOps

Solutions certification exam.

The certification exam is for DevOps professionals. Combine people, processes, and technologies to continuously deliver valuable products and services that meet end-user needs and business goals. DevOps professionals streamline delivery by optimizing practices, improving communications and collaboration, and creating automation.

They design and implement application code and infrastructure strategies that allow continuous integration, testing, delivery, monitoring, and feedback.

Exam candidates must be proficient with Agile practices. They must be familiar with Azure administration, development and experts in at least one of these areas.

DevOps professionals must design and implement DevOps practices for version control, compliance, infrastructure as code, configuration management, build, release, and testing-using Azure technologies.

There are five domain areas.

AZ-400 Domain Area	Weight
Configure Processes and Communications.	13%
Design and Implement Source Control.	19%
Design and Implement Build and Release Pipelines.	42%
Develop a Security and Compliance Plan.	14%
Implement an Instrumentation Strategy.	13%

Learning objectives

After completing this module, students and professionals can:

- Plan for the transformation with shared goals and timelines.
- Select a project and identify project metrics and Key Performance Indicators (KPIs).
- Create a team and agile organizational structure.
- Design a tool integration strategy.
- Design a license management strategy (for example, Azure DevOps and GitHub users).
- Design a plan for end-to-end traceability from work items to working software.
- Design an authentication and access strategy.
- Design a strategy for integrating on-premises and cloud resources.

Prerequisites

Successful learners will have prior knowledge and understanding of the following:

- Cloud computing concepts include understanding PaaS, SaaS, and IaaS implementations.

- Azure administration and Azure development with proven expertise in at least one of these areas.
- Version control, Agile software development, and core software development principles. It would be helpful to have experience in an organization that delivers software.

If you're new to Azure and cloud computing, consider one of the following resources:

- Free online: Azure Fundamentals .
- Instructor-led course: AZ-900: Azure Fundamentals .

If you're new to Azure Administration, consider taking the:

- Free online: Prerequisites for Azure Administrators .
- Instructor-led courses: AZ-104: Microsoft Azure Administrator .

If you're new to Azure Developer, consider taking the:

- Free online: Create serverless applications .
- Instructor-led courses: AZ-204: Developing Solutions for Microsoft Azure and AZ-020: Microsoft Azure Solutions for AWS Developers .

You must create an Azure DevOps Organization and a Team Project for some exercises. If you don't have it yet, see the following:

- Create an organization - Azure DevOps .
- If you already have your organization created, use the Azure DevOps Demo Generator [https://azuredevopsdemogenerator.azurewebsites.net] and create a new Team Project called "Parts Unlimited" using the template "PartsUnlimited." Or feel free to create a blank project. See Create a project - Azure DevOps .

You must create a GitHub account at GitHub.com and a project for some exercises. If you don't have it yet, see the following:

- Join GitHub · GitHub
- If you already have your GitHub account, create a new repository Creating a new repository - GitHub Docs .

Continue

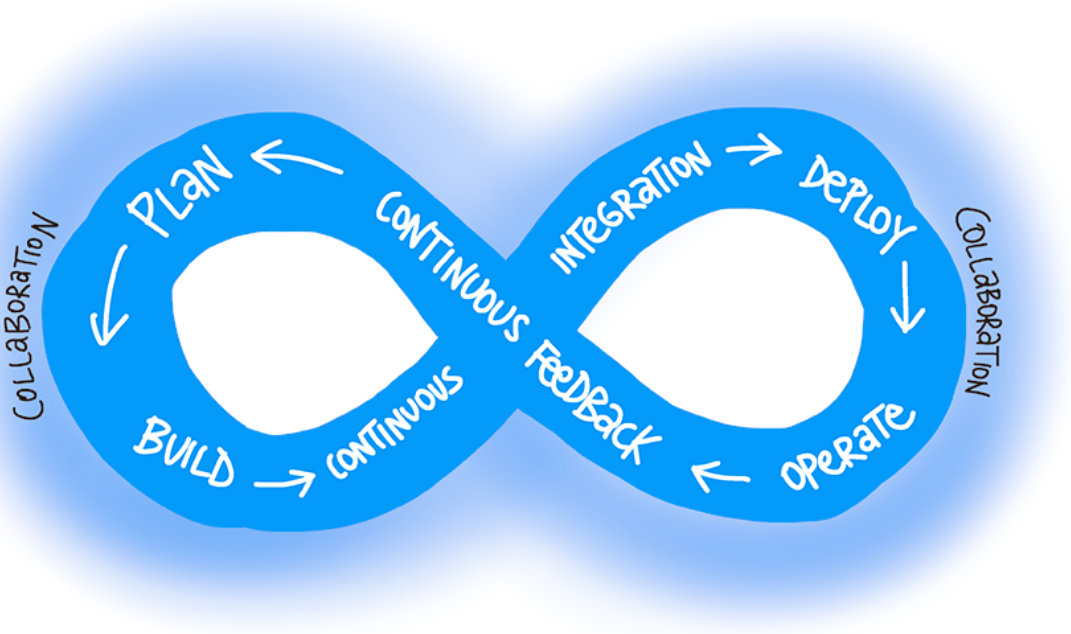
Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

What is DevOps?

Completed

- 3 minutes

The contraction of "Dev" and "Ops" refers to replacing siloed Development and Operations. The idea is to create multidisciplinary teams that now work together with shared and efficient practices and tools. Essential DevOps practices include agile planning, continuous integration, continuous delivery, and monitoring of applications. DevOps is a constant journey.



Understand your cycle time

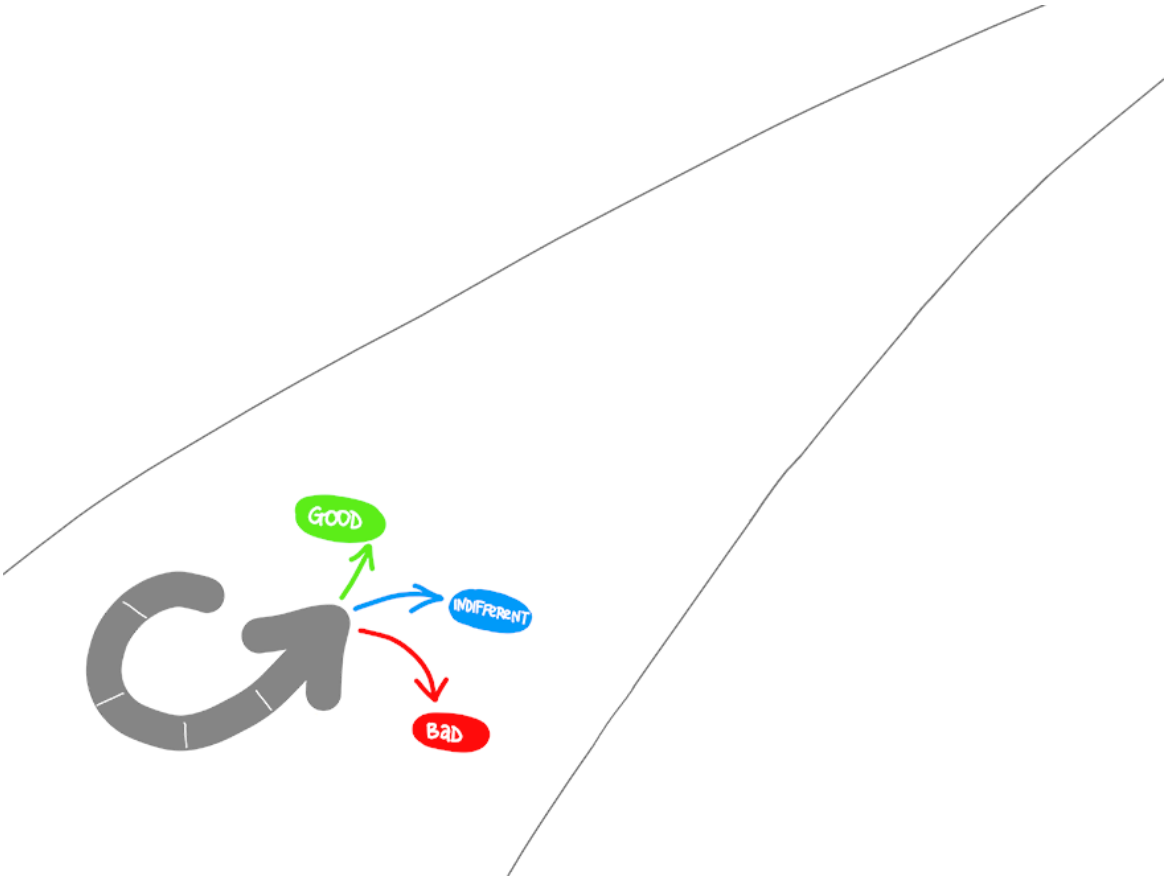
Let us start with a basic assumption about software development. We will describe it with the OODA (Observe, Orient, Decide, Act) loop. Originally designed to keep fighter pilots from being shot out of the sky, the OODA loop is an excellent way to think about staying ahead of your competitors. You start with observing business, market, needs, current user behavior, and available telemetry data. Then you orient with the enumeration of options for what you can deliver, perhaps with experiments. Next, you decide what to pursue, and you act by delivering working software to real users. You can see all occurring in some cycle time.

Become data-informed

We recommend you use data to inform what to do in your next cycle. Many experience reports tell us that roughly one-third of the deployments will have negative business results. Approximately one-third will have positive results, and one-third will make no difference. Fail fast on effects that do not advance the business and double down on outcomes that support the business. Sometimes the approach is called pivot or persevere.

Strive for validated learning

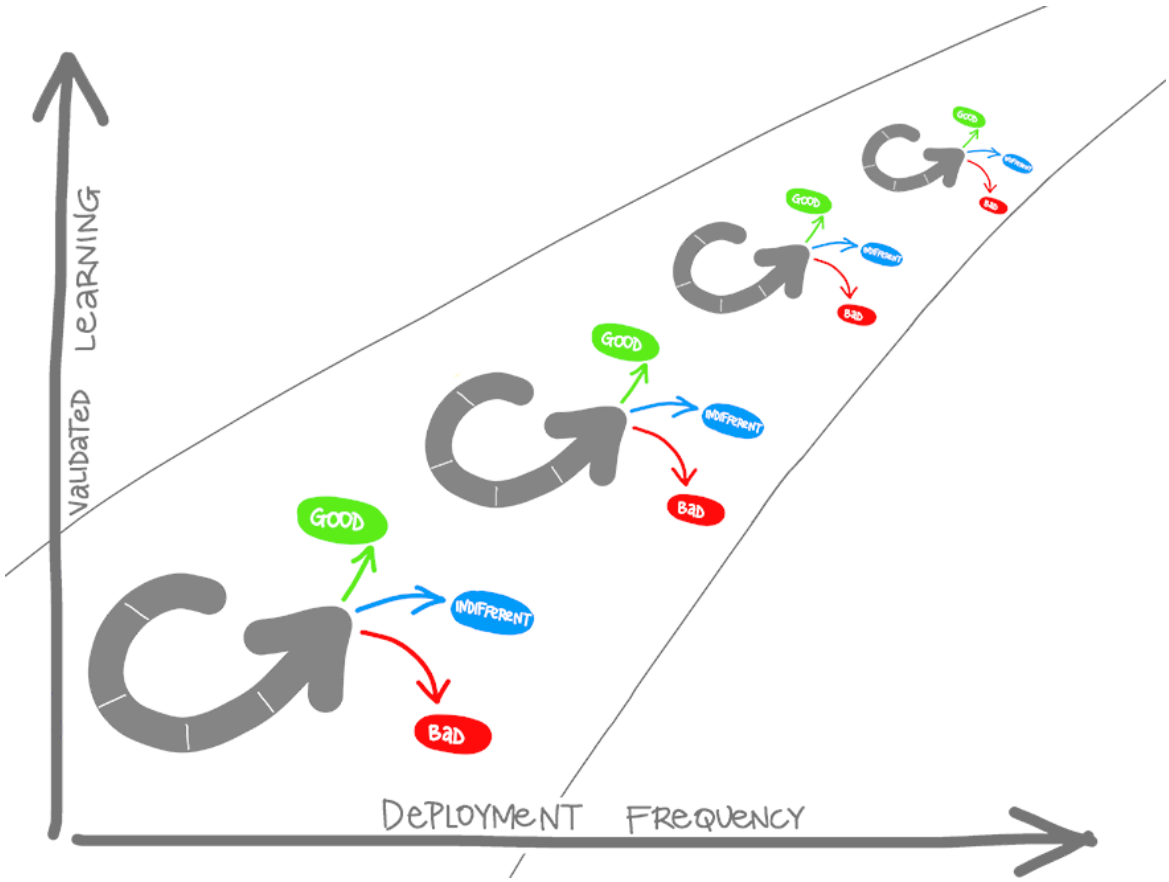
How quickly you can fail fast or double down is determined by your cycle time. Also, in how long that loop takes, or in lean terms. Your cycle time determines how quickly you can gather feedback to determine what happens in the next loop. The feedback that you collect with each cycle should be factual, actionable data. We call it validated learning.



Shorten your cycle time

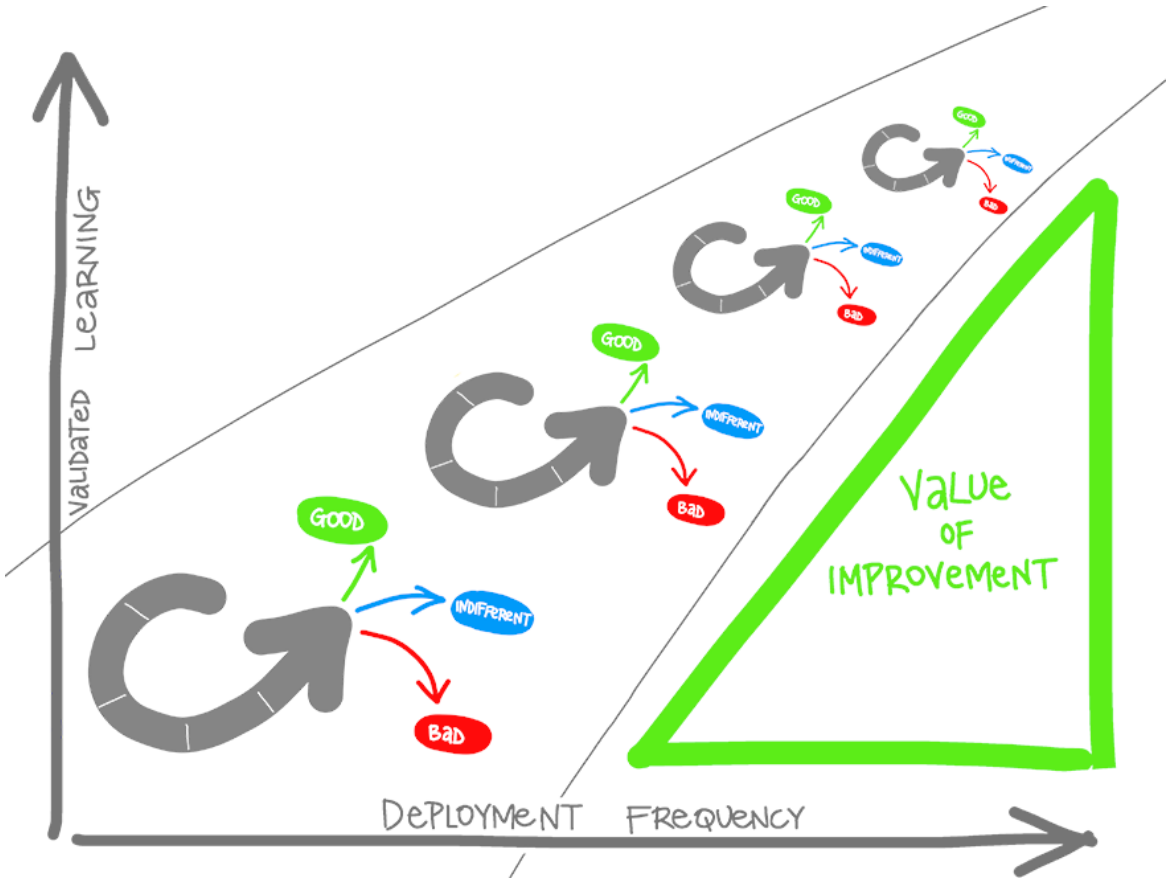
When you adopt DevOps practices:

- You shorten your cycle time by working in smaller batches.
- Using more automation.
- Hardening your release pipeline.
- Improving your telemetry.
- Deploying more frequently.



Optimize validated learning

The more frequently you deploy, the more you can experiment. The more opportunity you have to pivot or persevere and gain validated learning each cycle. This acceleration in validated learning is the value of the improvement. Think of it as the sum of progress that you achieve and the failures that you avoid.



Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

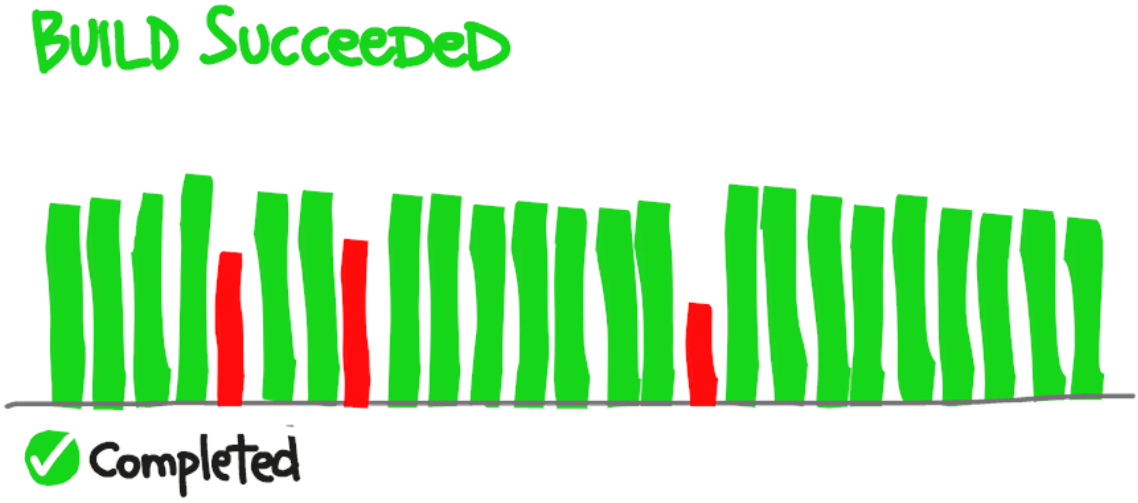
Explore the DevOps journey

Completed

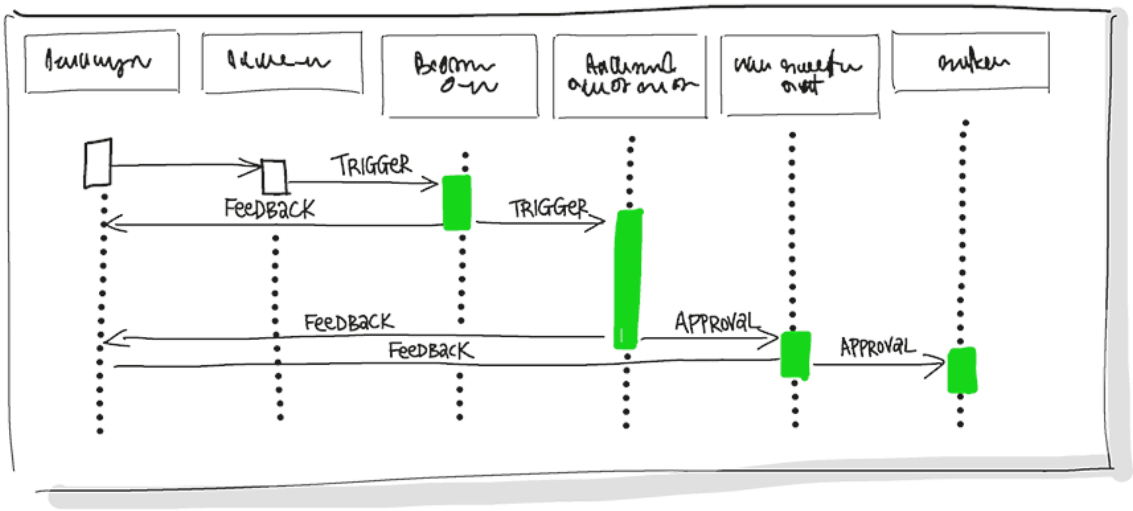
- 4 minutes

Remember, the goal is to shorten cycle time. Start with the release pipeline. How long does it take to deploy a change of one line of code or configuration? Ultimately, that is the brake on your velocity.

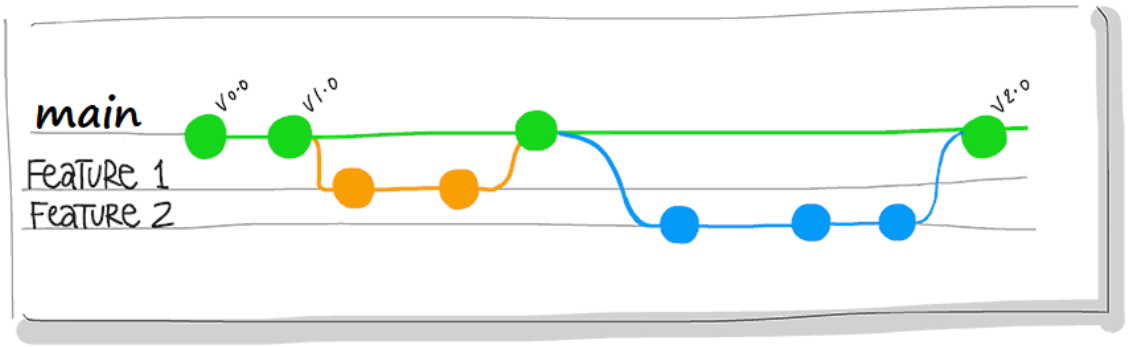
- Continuous Integration drives the ongoing merging and testing of code, leading to an early finding of defects. Other benefits include less time wasted fighting merge issues and rapid feedback for development teams.



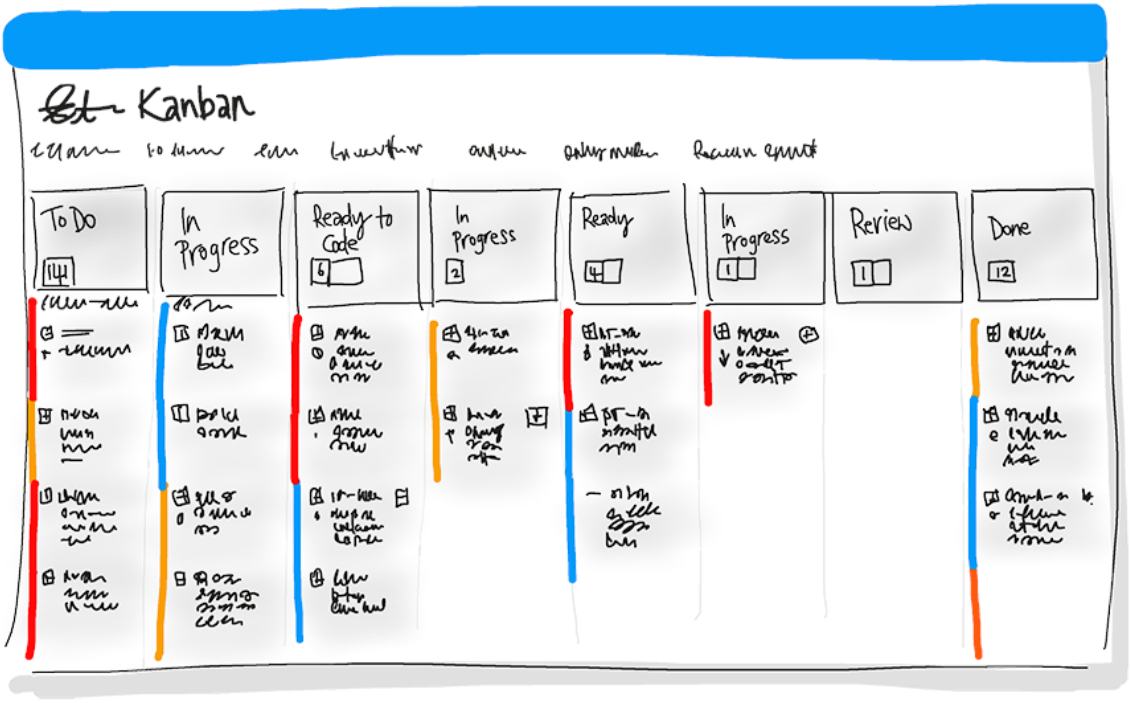
- Continuous Delivery of software solutions to production and testing environments helps organizations quickly fix bugs and respond to ever-changing business requirements.



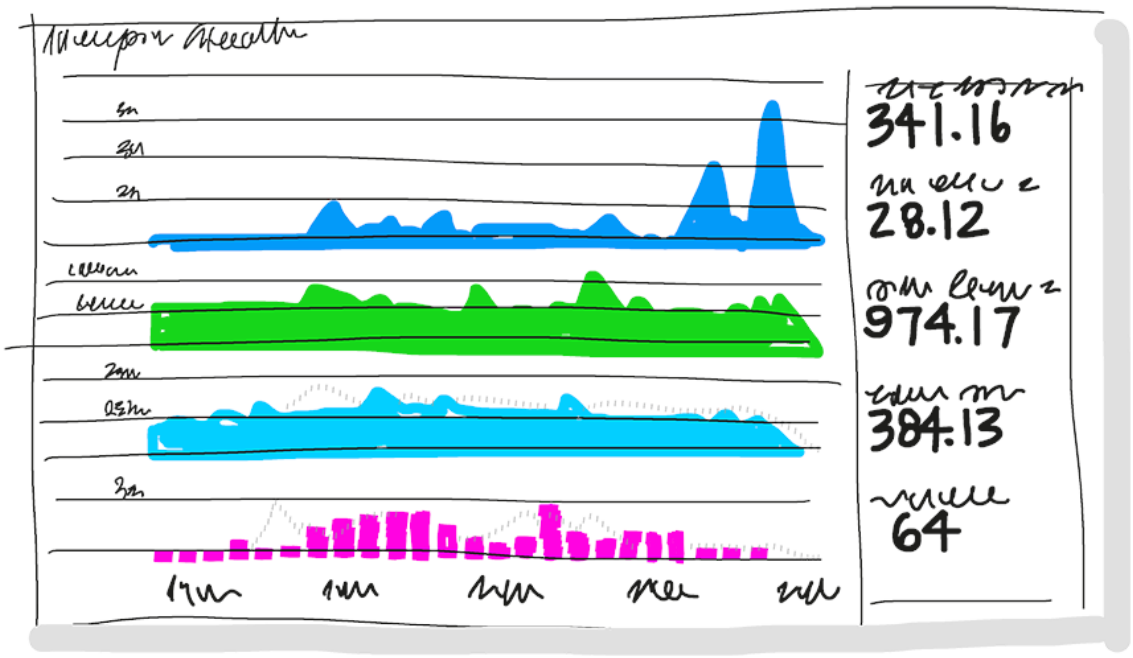
- Version Control, usually with a Git-based Repository, enables teams worldwide to communicate effectively during daily development activities. Also, integrate with software development tools for monitoring activities such as deployments.



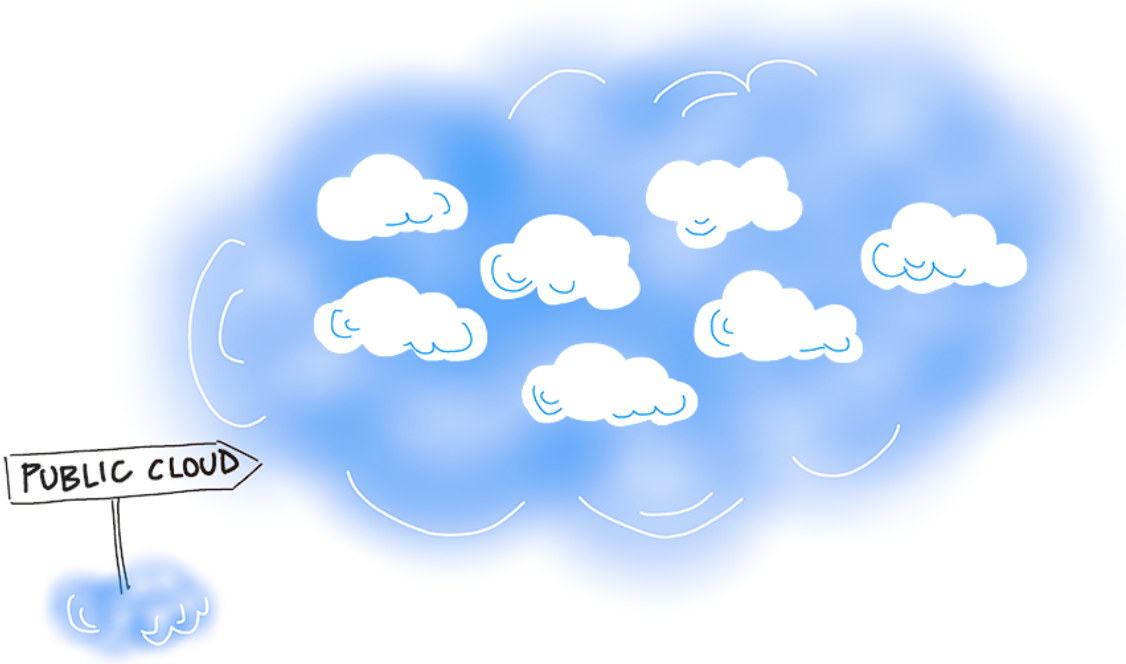
- Use Agile planning and lean project management techniques to:
  - Plan and isolate work into sprints.
  - Manage team capacity and help teams quickly adapt to changing business needs.
  - A DevOps Definition of Done is working software collecting telemetry against the intended business goals.



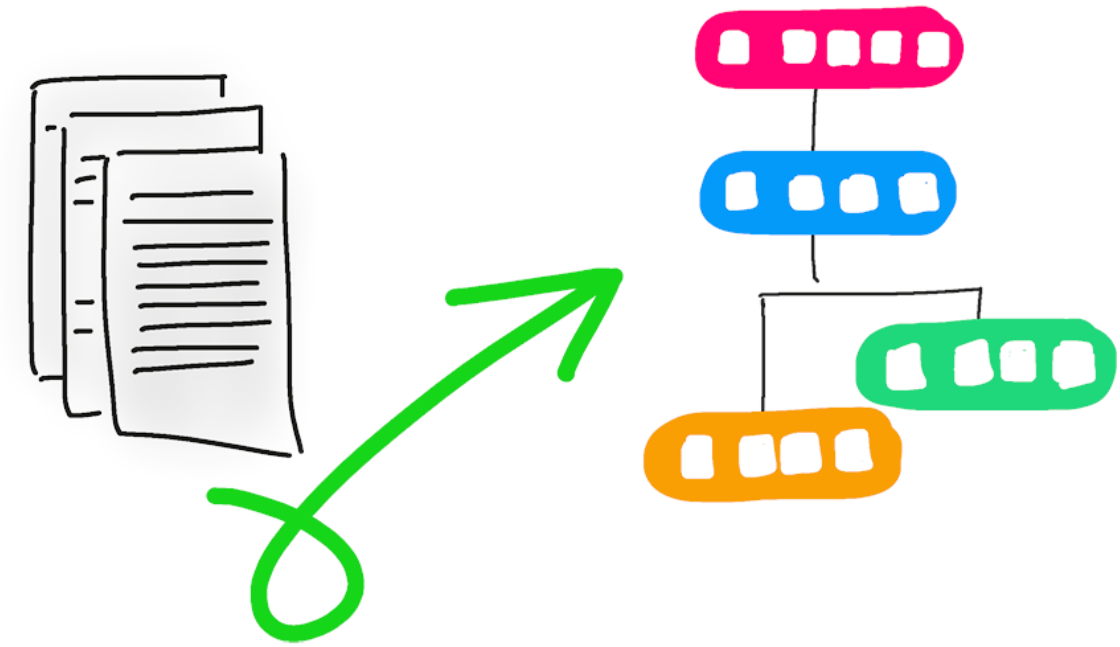
- Monitoring and Logging of running applications. Including production environments for application health and customer usage. It helps organizations create a hypothesis and quickly validate or disprove strategies. Rich data is captured and stored in various logging formats.



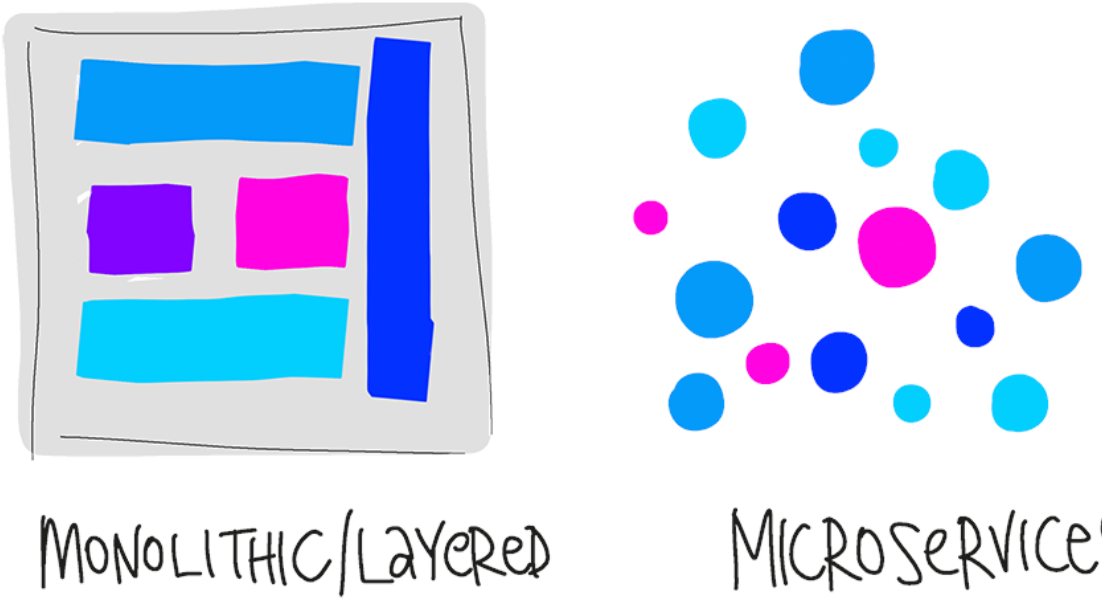
- Public and Hybrid Clouds have made the impossible easy. The cloud has removed traditional bottlenecks and helped commoditize Infrastructure. You can use Infrastructure as a Service (IaaS) to lift and shift your existing apps or Platform as a Service (PaaS) to gain unprecedented productivity. The cloud gives you a data center without limits.



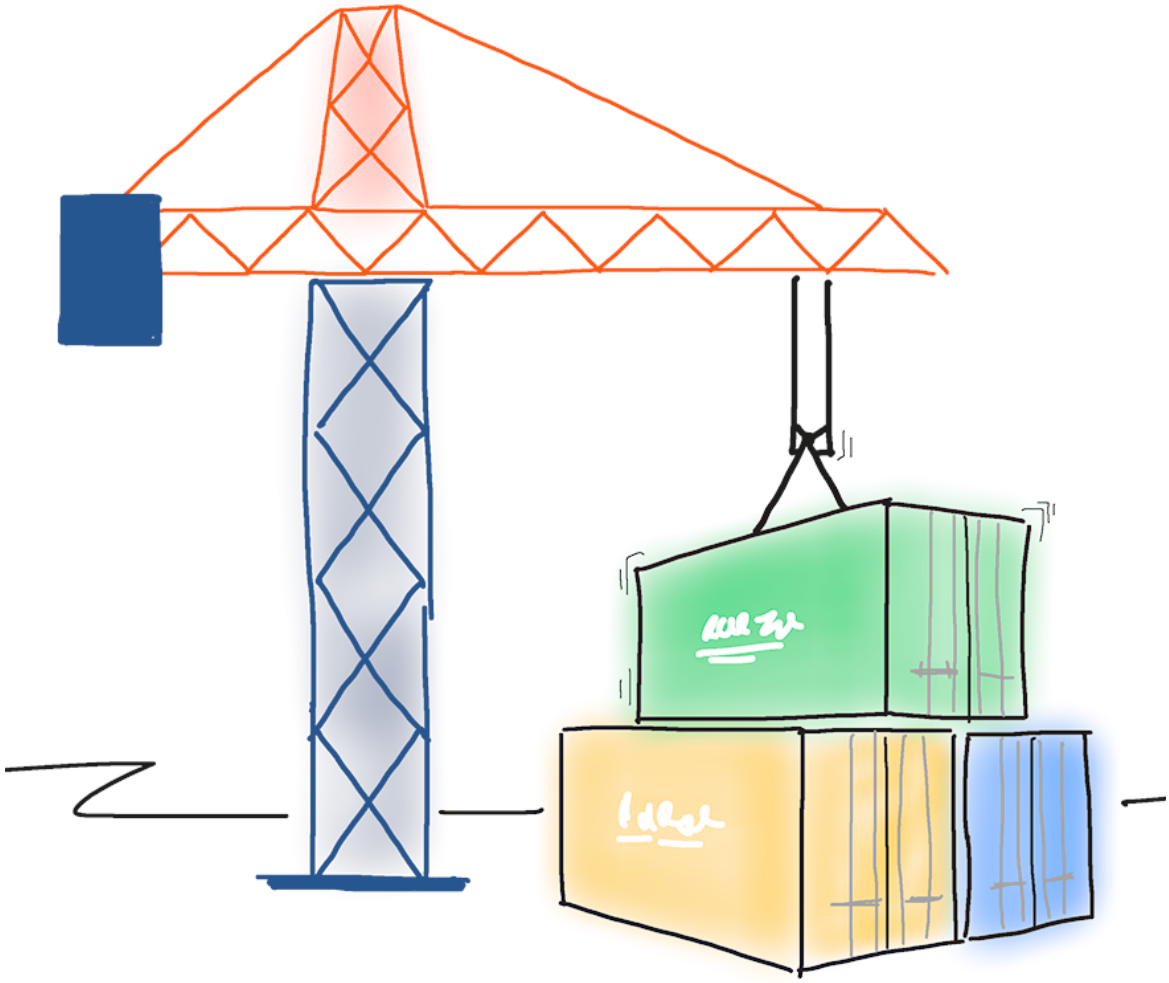
- Infrastructure as Code (IaC): Enables the automation and validation of the creation and teardown of environments to help deliver secure and stable application hosting platforms.



- Use Microservices architecture to isolate business use cases into small reusable services that communicate via interface contracts. This architecture enables scalability and efficiency.



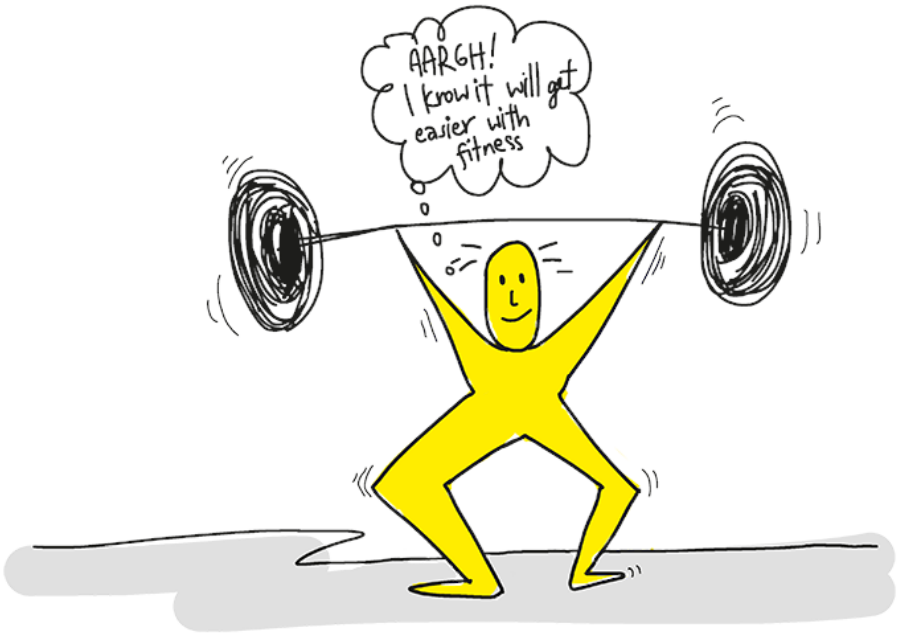
- Containers are the next evolution in virtualization. They're much more lightweight than virtual machines, allow much faster hydration, and easily configure files.



DevOps may hurt at first.

If it hurts, do it more often. Adopting new practices like going to the gym is likely to hurt first. The more you exercise the new techniques, the easier they'll become.

Like training at the gym, where you first exercise large muscles before small muscles, adopt practices that have the most significant impact. Cross-train to develop synergy.



Note

The source article defines DevOps .

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Identify transformation teams

Completed

- 2 minutes

Unless you're building an entirely new organization, one of the significant challenges with any DevOps Transformation Project is taking actions that conflict. At least in some way with ongoing business states.

The first challenge is the availability of staff. Suppose the staff members leading the transformation project are also involved in existing day-to-day work within the organization.

It will be challenging to focus on the transformation when their current role directly impacts customer outcomes.

We all know desperate situations involving customers will always win over a long-term project, like DevOps Transformations.

Another issue will be how the organization operates—implementing existing processes and procedures to support current business outcomes.

The disruption required for a true DevOps Transformation will usually challenge existing processes and procedures. Doing that is often difficult.

In the book "Beyond the Idea: How to Execute Innovation," Dr. Vijay Govindarajan and Dr. Chris Trimble noted when successful, they have researched what is involved in allowing Innovation to occur in organizations.

It has often been despite the existing organizational processes. Concluding it only works by creating a separate team to pursue the transformation.

For DevOps transformations, the separate team should be composed of staff members. Focused on and measured the transformation outcomes and not involved in the day-to-day operational work. The team might also include external experts that can fill the knowledge gaps—also helping to advise on processes that are new to the existing staff members.

Ideally, the staff members recruited should already be well-regarded throughout the organization. They should offer a broad knowledge base to think outside the box as a group.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Explore shared goals and define timelines

Completed

- 2 minutes

Explore shared goals

These outcomes should include specific, measurable targets 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.

Note

One of the aims of DevOps is to provide more excellent customer value, so outcomes should have a customer value focus.

Define timelines for goals

Measurable goals also need to have timelines. While it is easy to set longer-term goals, it is also easy to put off work when you do not require it for a while.

It is essential to have an ongoing series of short-term goals. Overall, projects should have timelines that span anywhere from a few months to a year or two in any DevOps transformation project.

Every few weeks, the improvements should be clear and measurable. Ideally, evident to the organization or its customers.

The timeline should not be too short and should always be challenging yet achievable. A review should occur after each short-term goal to help plan the next one.

There are several advantages of the shorter timelines:

- It is easier to change plans or priorities when necessary.
- The reduced delay between doing work and getting feedback helps ensure that the learnings and feedback are incorporated quickly.
- It is easier to keep organizational support when positive outcomes are clear.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Knowledge check

Completed

- 4 minutes

Choose the best response for each question. Then select **Check your answers** .

Check your knowledge

1.

Which of the following choices best describes DevOps?

DevOps is the role of who manages source control, pipelines, and monitor environments to continue delivering value to the software project.

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

DevOps is the new process of creating continuous delivery and continuous integration for software projects.

2.

Which of the following choices drives the ongoing merging and testing of code that leads to finding defects early?

Continuous Integration.

Continuous Delivery.

Continuous Feedback.

3.

Which of the following choices is a practice that enables the automated creation of environments?

Infrastructure as a Service (IaaS).

Infrastructure as Code (IaC).

Software as a Service (SaaS).

Check your answers

You must answer all questions before checking your work.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Summary

Completed

- 2 minutes

This module explored the key areas that organizations must apply to start their DevOps transformation Journey, changing the team's mindset, and defining timelines and goals.

You learned how to describe the benefits and usage of:

- Understand what DevOps is and the steps to accomplish it.
- Identify teams to implement the process.
- Plan for the transformation with shared goals and timelines.
- Plan and define timelines for goals.

Learn more

- Donovan Brown | What is DevOps?
- What is DevOps? - Azure DevOps | Microsoft Docs
- Getting started with GitHub - GitHub Docs
- View of features and epics on the Feature Timeline - Azure DevOps | Microsoft Docs
- Plan and track work in Azure Boards with Basic or Agile processes - Azure Boards | Microsoft Docs
- Agile Manifesto for Software Development | Agile Alliance
- 12 Principles Behind the Agile Manifesto | Agile Alliance

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Choose the right project

Introduction

Completed

- 2 minutes

Are you interested in implementing DevOps practices but are not sure what it can do for you?

This module helps organizations decide how to apply the DevOps process and tools to minimize initial resistance.

Learning objectives

After completing this module, students and professionals can:

- Understand different projects and systems to guide the journey.
- Select a project to start the DevOps transformation.
- Identify groups to minimize initial resistance.
- Identify project metrics and Key Performance Indicators (KPIs).

Prerequisites

- Understanding of what DevOps is and its concepts.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Explore greenfield and brownfield projects

Completed

- 2 minutes

The terms greenfield and brownfield have their origins in residential and industrial building projects.

A greenfield project is one done on a green field, undeveloped land. A brownfield project is done on the used ground for other purposes.

Because of the land use that has once occurred, there could be challenges reusing the land. Like existing buildings, some would be obvious but less obvious, like polluted soil.

Applied to software or DevOps Projects

The same terms are routinely applied to software projects and commonly describe DevOps Projects.

On the surface, it can seem that a greenfield DevOps project would be easier to manage and to achieve success.

- There was no existing codebase.
- No existing team dynamics of politics. Possibly no current, rigid processes.

A common misconception is that DevOps is only for greenfield projects and suits startups best. However, DevOps can also succeed with brownfield projects.

The beauty of these projects is that there's often a large gap between customer expectations and delivery.

The teams involved may well realize that the status quo needs to change. They've lived the challenges and the limitations associated with what they're currently doing.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Decide when to use greenfield and brownfield projects

Completed

- 3 minutes

When starting a DevOps transformation, you might need to choose between greenfield and brownfield projects.

There's a common misconception that we need to demystify that DevOps suits greenfield projects better than brownfield projects.

Greenfield projects

A greenfield project will always appear to be a more accessible starting point. A blank slate offers the chance to implement everything the way that you want.

You might also have a better chance of avoiding existing business processes that don't align with your project plans.

Suppose current IT policies don't allow the use of cloud-based infrastructure. In that case, the project might be qualified for entirely new applications designed for that environment from scratch.

For example, you can sidestep internal political issues that are well entrenched.

Brownfield projects

Usually, brownfield projects come with:

- The baggage of existing codebases.
- Existing teams.
- A significant amount of technical debt.

But, they can still be ideal projects for DevOps transformations.

When your teams spend large percentages of their time just maintaining existing brownfield applications, you have limited ability to work on new code.

It's essential to find a way to reduce that time and to make software release less risky. A DevOps transformation can provide that.

The limitations will have often worn down the existing team members. For example, they're working in the past and aren't keen to experiment with new ideas.

The system is often crucial for organizations. It might also be easier to gain more robust management buy-in for these projects because of the potential benefits delivered.

Management might also have a stronger sense of urgency to point brownfield projects in an appropriate direction when compared to greenfield projects that don't currently exist.

Take the first step

Eventually, the goal will be to evolve your entire organization. In looking to take the first step, many organizations start with a greenfield project and then move on.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Decide when to use systems of record versus systems of engagement

Completed

- 2 minutes

When selecting systems as candidates for starting a DevOps transformation, it is necessary to consider the types of systems that you operate.

Some researchers suggest that organizations often use Bimodal IT, a practice of managing two separate, coherent modes of IT delivery - one focused on stability and predictability and the other on agility.

Systems of record

Systems that provide the truth about data elements are often-called systems of record. These systems have historically evolved slowly and carefully. For example, it is crucial that a banking system accurately reflects your bank balance. Systems of record emphasize accuracy and security.

Systems of engagement

Many organizations have other systems that are more exploratory. These often use experimentation to solve new problems. Systems of engagement are modified regularly. Usually, it is a priority to make quick changes over ensuring that the changes are correct.

There is a perception that DevOps suits systems of engagement more than systems of record. The lessons from high-performing companies show that is not the case.

Sometimes, the criticality of doing things right with a system of record is an excuse for not implementing DevOps practices.

Worse, given the way that applications are interconnected, an issue in a system of engagement might end up causing a problem in a system of record anyway.

Both types of systems are great. At the same time, it might be easier to start with a system of engagement when first beginning a DevOps Transformation.

DevOps practices apply to both types of systems. The most significant outcomes often come from transforming systems of record.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Identify groups to minimize initial resistance

Completed

- 2 minutes

Not all staff members within an organization will be receptive to the change required for a DevOps transformation.

In discussions around continuous delivery, we usually categorize users into three general buckets:

- **Canary** users voluntarily test bleeding edge features as soon as they're available.
- **Early adopters** who voluntarily preview releases, considered more refined than the code that exposes canary users.
- **Users** who consume the products after passing through canary and early adopters.

It's essential to find staff members keen to see new features as soon as they're available and highly tolerant of issues when choosing Canary.

Early adopters have similar characteristics to the Canaries. They often have work requirements that make them less tolerant of issues and interruptions to work.

While development and IT operations staff might generally be less conservative than users.

The staff will also range from traditional to early adopters, and others happy to work at the innovative edge.

Ideal target improvements

It's also important to roll out changes incrementally. There is an old saying in the industry that any successful large IT system was previously a successful small IT system.

Large-scale systems rolled out all at once have an abysmal record of success. Most fail, no matter how much support management has provided.

When starting, it is essential to find an improvement goal that:

- It can be used to gain early wins.
- It is small enough to be achievable in a reasonable timeframe.
- Has benefits that are significant enough to be evident to the organization.

It allows constant learning from rapid feedback and recovering from mistakes quickly.

Note

The aim is to build a snowball effect where each new successful outcome adds to previous successful results. It will maximize the buy-in from all affected.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Identify project metrics and key performance indicators (KPIs)

Completed

- 3 minutes

We spoke earlier about the importance of shared goals. It was also agreed upon by team members that the goals needed to be specific, measurable, and time-bound.

It is essential to establish (and agree upon) appropriate metrics and Key Performance Indicators (KPIs) to ensure these goals are measurable.

While there is no specific list of metrics and KPIs that apply to all DevOps Projects, the following are commonly used:

Faster outcomes

- **Deployment Frequency** . Increasing the frequency of deployments is often a critical driver in DevOps Projects.
- **Deployment Speed** . It is necessary to reduce the time that they take.
- **Deployment Size** . How many features, stories, and bug fixes are being deployed each time?
- **Lead Time** . How long does it take from the creation of a work item until it is completed?

Efficiency

- **Server to Admin Ratio** . Are the projects reducing the number of administrators required for a given number of servers?
- **Staff Member to Customers Ratio** . Is it possible for fewer staff members to serve a given number of customers?
- **Application Usage** . How busy is the application?
- **Application Performance** . Is the application performance improving or dropping? (Based upon application metrics)?

Quality and security

- **Deployment failure rates** . How often do deployments (or applications) fail?
- **Application failure rates** . How often do application failures occur, such as configuration failures, performance timeouts, and so on?
- **Mean time to recover** . How quickly can you recover from a failure?
- **Bug report rates** . You do not want customers finding bugs in your code. Is the amount they are seeing increasing or lowering?
- **Test pass rates** . How well is your automated testing working?
- **Defect escape rate** . What percentage of defects are being found in production?
- **Availability** . What percentage of time is the application truly available for customers?
- **Service level agreement achievement** . Are you meeting your service level agreements (SLAs)?
- **Mean time to detection** . If there is a failure, how long does it take for it to be detected?

Culture

- **Employee morale** . Are employees happy with the transformation and where the organization is heading? Are they still willing to respond to further changes? This metric can be challenging to measure but is often done by periodic, anonymous employee surveys.
- **Retention rates** . Is the organization losing staff?

Note

It is crucial to choose metrics that focus on specific business outcomes and achieve a return on investment and increased business value.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Knowledge check

Completed

- 4 minutes

Choose the best response for each question. Then select **Check your answers** .

Check your knowledge

1.

In which of the following choices would you find large amounts of technical debt?

○

Greenfield project.

○

Brownfield project.

○

Bluefield project.

2.

Which of the following choices would a system that manages inventory in a warehouse be considered?

System of Record.

System of Engagement.

System of History.

3.

Which of the following choices are the categorized user groups most adopted in Continuous Delivery?

Canaries, Early adopters, and Users.

Alpha and Beta Users.

Blue and Green Users.

Check your answers

You must answer all questions before checking your work.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Summary

Completed

- 2 minutes

In this module, you learned how to decide how to apply the DevOps process and tools to minimize initial resistance.

Also, how to identify teams, plan for a DevOps transformation culture, and define timelines for your goals.

You learned how to describe the benefits and usage of:

- Understand what DevOps is and the steps to accomplish it.
- Identify teams to implement the process.
- Plan for the transformation with shared goals and timelines.
- Plan and define timelines for goals.

Learn more

- Greenfield project - Wikipedia
- Brownfield (software development) - Wikipedia
- About teams & Agile tools | Microsoft Docs
- Best practices for Agile project management | Microsoft Docs

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Describe team structures

Introduction

Completed

- 1 minute

Starting to apply Agile practices in your company is not easy. It demands collaboration from your teams, stakeholder support, and training.

This module explores agile development practices and helps to define and to configure teams and tools for collaboration.

Learning objectives

After completing this module, students and professionals can:

- Understand agile practices and principles of agile development.
- Create a team and agile organizational structure.
- Identify ideal DevOps team members.
- Select and configure tools for collaboration.

Prerequisites

- Understanding of what DevOps is and its concepts.
- Familiarity with Agile software development and core software development principles is helpful but is not necessary.
- Beneficial to have experience in an organization that delivers software.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Explore agile development practices

Completed

- 3 minutes

Waterfall

Traditional software development practices involve:

- Determining a problem.
- Analyzing the requirements.
- Building and testing the required code.
- The delivery outcome to users.

Usually, all refer to as a waterfall approach.

The waterfall model follows a sequential order. A project development team only moves to the next development phase or testing if the previous step is completed successfully.

It's what an engineer would do when building a bridge or a building. So, it might seem appropriate for software projects as well.

However, the waterfall methodology has some drawbacks. One relates to the customer requirements.

For example, It doesn't matter if the customer requirements are defined accurately at the start of a project.

Usually, the project takes a long time, and the outcome may no longer match the customer's needs.

There's a real challenge with gathering customer requirements in the first place.

Taking a long time to deliver something would often be different from what the customer needs, even if you built exactly what the customer asked.

Customers often don't know what they want until they see it or can't explain what they need.

Agile

By comparison, Agile methodology constantly emphasizes adaptive planning and early delivery with continual improvement.

Rather than restricting development to rigid specifications, it encourages rapid and flexible responses to changes as they occur.

In 2001, highly regarded developers published a manifesto for Agile software development.

They said that:

- Development needs to favor individuals and interactions over processes and tools.
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation.
- Respond to changes over following a plan.

Agile software development methods are based on releases and iterations:

- One release might consist of several iterations.
- Each iteration is like a small independent project.
- After being estimated and prioritization:
    - Features, bug fixes, enhancements, and refactoring width are assigned to a release.
    - And then assigned again to a specific iteration within the release, generally on a priority basis.
- At the end of each iteration, there should be tested working code.
- In each iteration, the team must focus on the outcomes of the previous iteration and learn from them.

Having teams focused on shorter-term outcomes is that teams are also less likely to waste time over-engineering features. Or allowing unnecessary scope creep to occur.

Agile software development helps teams keep focused on business outcomes.

Comparison of the waterfall and agile methodologies

Waterfall	Agile
Divided into distinct phases.	Separates the project development lifecycle into sprints.
It can be rigid.	Known for flexibility.
All project development phases, such as design, development, and test, are completed once.	It follows an iterative development approach so that each phase may appear more than once.
Define requirements at the start of the project with little change expected.	Requirements are expected to change and evolve.
Focus on completing the project.	Focus on meeting customers' demands.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Explore principles of agile development

Completed

- 2 minutes

The Agile Alliance says its mission is to support people who explore and apply agile values and principles. Also, practices to make building software solutions more effective, humane, and sustainable.

They have published a Manifesto for Agile Software Development .

And from the publication, they have distilled the 12 Principles Behind the Agile Manifesto .

- Our highest priority is to satisfy the customer through the early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of months to a couple of weeks, with a preference for a shorter timescale.
- Businesspeople and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity - the art of maximizing the amount of work not done - is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- The team regularly reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Define organization structure for agile practices

Completed

- 3 minutes

For most organizations, reorganizing to be agile is difficult. It requires a mind-shift and a culture-shift that challenges many existing policies and processes within the organization.

Good governance in organizations, particularly in large organizations, often leads to many relatively rigid rules, operating structures, and methods. It also tends to avoid a broad delegation of authority.

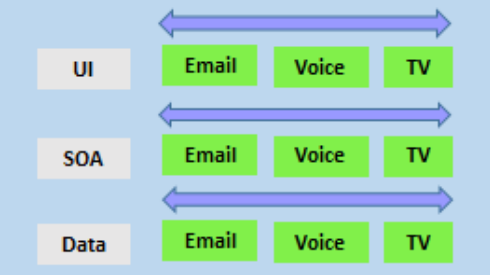
While most large organizations have not moved to an agile structure, most are now experimenting with doing so.

Their business environments are volatile and complex, and they have seen the limitations of their current systems, mainly an inability to cope with change quickly.

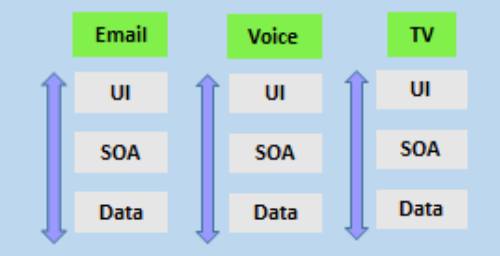
They realize that it is common today for long-term established businesses and their industries to be disrupted by startups.

Horizontal vs. vertical teams

Traditionally, horizontal team structures divide teams according to the software architecture. In this example, the teams have been divided into the user interface, service-oriented architecture, and data teams:

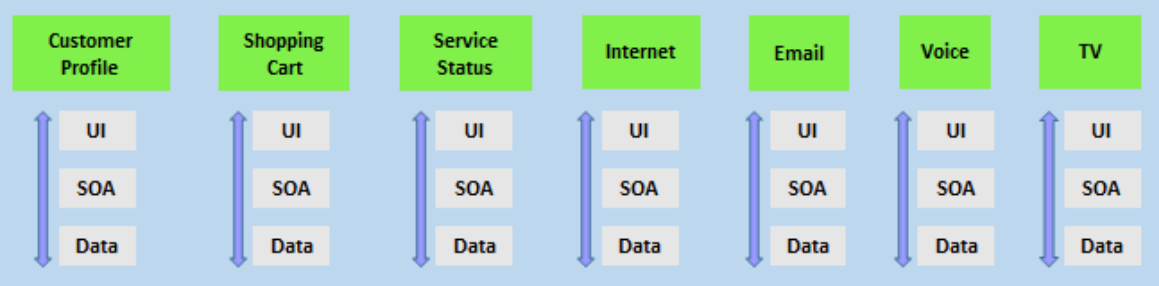


By comparison, vertical team structures span the architecture and are aligned with skillsets or disciplines:



Vertical teams have been shown to provide more good outcomes in Agile projects. Each product must have an identified owner.

Another key benefit of the vertical team structure is that scaling can occur by adding teams. In this example, feature teams have been created rather than just project teams:



Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Explore ideal DevOps team members

Completed

- 2 minutes

For a successful DevOps transformation, the aim is to find team members with the following characteristics:

- They already think there is a need to change.
- They have previously shown an ability to innovate.
- They are already well respected within the organization.
- They have a broad knowledge of the organization and how it operates.
- Ideally, they already believe that DevOps practices are what is needed.

Mentoring team members on agile practices

While it's desirable to have formal agile training for staff members, no matter how good any agile course is, there is a world of difference between learning a concept within a few days and putting it into practice.

When they first start an agile transformation, many teams hire external coaches or mentors.

Agile coaches help teams or individuals to adopt agile methods or to improve the current techniques and practices.

They must be agents of change by helping people understand how they work and encouraging them to adopt new approaches.

Agile coaches typically work with more than one team and remove any roadblocks from inside or outside the organization.

This work requires various skills, including coaching, mentoring, teaching, and making easier. Agile coaches must be both trainers and consultants.

There is more than one type of agile coach.

- Some coaches are technical experts who aim to show staff members how to apply specific concepts—for example, test-driven development and continuous integration or deployment.
    - These coaches might do peer programming sessions with staff members.
- Other coaches are focused on agile processes, determining requirements, and managing work activities.
    - They might help how to run effective stand-up and review meetings.
    - Some coaches may themselves act as scrum masters.
        - They might mentor staff in how to fill these roles.

Over time, though, team members need to develop an ability to mentor each other. Teams should aim to be self-organizing. Team members are often expected to learn as they work and to acquire skills from each other. To make it effective, though, the work itself needs to be done collaboratively, not by individuals working by themselves.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Enable in-team and cross-team collaboration

Completed

- 4 minutes

Effective collaboration is critical for well-functioning Agile teams. Enabling it requires cultural changes, cross-functional team collaboration, and tooling.

Cultural changes

Over recent decades, offices have often become open spaces with few walls. At the time of writing, a significant shift to working from home started as a response to the pandemic. Both situations can limit collaboration, and ambient noise and distractions often reduce productivity. Staff tends to work better when they have comfortable working environments. Defined meeting times and locations let teams choose when they want to interact with others.

Asynchronous communication should be encouraged, but there should not be an expectation that all communications will be responded to urgently. Staff should focus on their primary tasks without feeling like they are being left out of important decisions.

All meetings should have strict timeframes and, more importantly, have a plan. If there is no plan, there should be no meeting.

As it is becoming harder to find the required staff, great teams will be as comfortable with remote or work-from-home as with workers in the office.

To be successful, though, collaboration via communication should become part of the organization's DNA.

Staff should be encouraged to communicate openly and frankly. Learning to deal with conflict is essential for any team, as there will be disagreements at some point. Mediation skills training would be helpful.

Cross-functional teams

Team members need good collaboration. It is also essential to have a great partnership with wider teams to bring people with different functional expertise together to work toward a common goal.

Often, there will be people from other departments within an organization.

Faster and better innovation can occur in these cross-functional teams.

People from different areas of the organization will have different views of the same problem and are more likely to come up with alternate solutions to problems or challenges. Existing entrenched ideas are more likely to be challenged.

Cross-functional teams can also minimize turf wars within organizations. The more widely a project appears to have ownership, the easier it will be to be widely accepted. Bringing cross-functional teams together also helps to spread knowledge across an organization.

Recognizing and rewarding collective behavior across cross-functional teams can also help to increase team cohesion.

Collaboration tooling

Agile teams commonly use the following collaboration tools:

Teams (Microsoft) : A group chat application from Microsoft. It provides a combined location with workplace chat, meetings, notes, and storage of file attachments. A user can be a member of many teams.

Slack : A commonly used tool for collaboration in Agile and DevOps teams. From a single interface, it provides a series of separate communication channels that can be organized by project, team, or topic. Conversations are kept and are searchable. It is straightforward to add both internal and external team members. Slack integrates with many third-party tools like GitHub for source code and DropBox for document and file storage.

Jira : A commonly used tool for planning, tracking, releasing, and reporting.

Asana : A standard tool designed to keep team plans, progress, and discussions in a single place. It has strong capabilities around timelines and boards.

Glip : An offering from Ring Central that provides chat, video, and task management.

Other standard tools with collaboration offerings include ProofHub, RedBooth, Trello, DaPulse, and many others.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Select tools and processes for agile practices

Completed

- 2 minutes

While developing agile methods does not require specific tooling, tools can often enhance the achieved outcomes.

It is essential to realize that the vital tool for agile development is the process itself.

Become familiar with the procedures you need to follow before working out how to implement tools. Several categories of tools are commonly used.

Physical tools

Not all tools need to be digital tools. Many teams use whiteboards to collaborate on ideas, index cards for recording stories, and sticky notes for moving around tasks.

Even when digital tools are available, it might be more convenient to use these physical tools during stand-up and other meetings.

Collaboration tools

We described collaboration tools in the previous topic.

Project management tools

These tools usually include:

- Project planning and execution monitoring abilities (including how to respond to impediments).
- Automation for stand-up meetings.
- Management and tracking of releases.
- A way to record and work with the outcomes of retrospectives.
- Many include Kanban boards and detailed sprint planning options.

These tools will also provide detailed visualizations, often as a graphic dashboard that shows team progress against assigned goals and targets.

Some tools integrate directly with code repositories and CI/CD tools and add code-related metrics, including quality metrics and direct support for code reviews.

As a complete CI/CD system, we have Azure DevOps and GitHub that includes:

- Flexibility in Kanban boards.
- Traceability through Backlogs.
- Customizability in dashboards.
- Built-in scrum boards.
- Integrability directly with code repositories.
- Code changes can be linked directly to tasks or bugs.

Apart from Azure DevOps and GitHub, other standard tools include:

- Jira.
- Trello.
- Active Collab.
- Agilo for Scrum.
- SpiraTeam.
- Icescrum.
- SprintGround.
- Gravity.
- Taiga.
- VersionOne.
- Agilean.
- Wrike.
- Axosoft.
- Assembla.
- PlanBox.
- Asana.
- Binfire.
- Proggio.
- VivifyScrum, and many others.

Screen recording tools

It might seem odd to add screen recording tools to this list. Still, they are beneficial when:

- Work with remote team members.
- Recording bugs in action.
- Building walkthroughs and tutorials that demonstrate actual or potential features.

A screen recorder is built into Windows, but other common ones include SnagIt, Camtasia, OBS, and Loom.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Knowledge check

Completed

- 4 minutes

Choose the best response for each question. Then select **Check your answers** .

Check your knowledge

1.

An Agile tool manages and visualizes work by showing tasks moving from left to right across columns representing stages. What is this tool commonly called?

○

Backlog.

○

Kanban Board.

○

Delivery Plans.

2.

Which of the following choices isn't describing processes and methodologies correctly?

○

It is required to implement Waterfall to evolve and implement Agile.

○

It is possible to start with Waterfall and evolve to Agile.

○

Agile is the best and most used methodology and the best to start your project.

3.

Which of the following choices isn't an Agile/Project management tool?

○

Azure DevOps.

○

GitHub.

○

Camtasia.

Check your answers

You must answer all questions before checking your work.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Summary

Completed

- 1 minute

This module explored agile development practices and helped define and configure teams and tools for collaboration.

You learned how to describe the benefits and usage of:

- Understand agile practices and principles of agile development.
- Create a team and agile organizational structure.
- Identify ideal DevOps team members.
- Select and configure tools for collaboration.

Learn more

- DevOps vs. Agile | Microsoft Azure .
- Best practices for Agile project management - Azure Boards | Microsoft Docs .
- Agile Manifesto for Software Development | Agile Alliance .
- Agile Board | Trello .
- Agile Alliance .
- 12 Principles Behind the Agile Manifesto | Agile Alliance .
- Jira | Issue & Project Tracking Software | Atlassian .

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Choose the DevOps tools

Introduction

Completed

- 1 minute

Do you currently use tools to manage daily work and plan for the future? It is essential when implementing DevOps if you want to increase your maturity level.

This module explores Azure DevOps and GitHub tools and helps organizations define their work management tool and licensing strategy.

Learning objectives

After completing this module, students and professionals can:

- Design a tool integration strategy.
- Design a license management strategy (for example, Azure DevOps and GitHub users).
- Design a strategy for end-to-end traceability from work items to working software.
- Design an authentication and access strategy.
- Design a strategy for integrating on-premises and cloud resources.

Prerequisites

- Understanding of what DevOps is and its concepts.
- Familiarity with DevOps tools is helpful but is not necessary.
- Beneficial to have experience in an organization that delivers software.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

What is Azure DevOps?

Completed

- 2 minutes

Azure DevOps is a Software as a service (SaaS) platform from Microsoft that provides an end-to-end DevOps toolchain for developing and deploying software.

It also integrates with the most-leading tools on the market and is an excellent option for orchestrating a DevOps toolchain.

What does Azure DevOps provide?

Azure DevOps includes a range of services covering the complete development life cycle.

- Azure Boards: agile planning, work item tracking, visualization, and reporting tool.
- Azure Pipelines: a language, platform, and cloud-agnostic CI/CD platform-supporting containers or Kubernetes.
- Azure Repos: provides cloud-hosted private git repos.
- Azure Artifacts: provides integrated package management with support for Maven, npm, Python, and NuGet package feeds from public or private sources.
- Azure Test Plans: provides an integrated planned and exploratory testing solution.

Also, you can use Azure DevOps to orchestrate third-party tools.

What if we are not a Microsoft / Microsoft .NET organization?

Azure DevOps is not focused on organizations that are end-to-end Microsoft or Windows.

Azure DevOps provides a platform that is:

- Flexible: you do not have to go 'all in' on Azure DevOps. It is possible to adopt each of the services independently and integrate them with your existing toolchain; most popular tools are supported.
- Cross-Platform: designed to work with any platform (Linux, macOS, and Windows). Or language (including Node.js, Python, Java, PHP, Ruby, C/C++, .NET, Android, and iOS apps). Azure DevOps is not aimed at organizations building and shipping on the Microsoft technology stack.
- Cloud Agnostic: continuous delivery is supported to AWS, GCP, and Azure.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

What is GitHub?

Completed

- 1 minute

GitHub is a Software as a service (SaaS) platform from Microsoft that provides Git-based repositories and DevOps tooling for developing and deploying software.

It has a wide range of integrations with other leading tools.

What does GitHub provide?

GitHub provides a range of services for software development and deployment.

- **Codespaces:** Provides a cloud-hosted development environment (based on Visual Studio Code) that can be operated from within a browser or external tools. Eases cross-platform development.
- **Repos:** Public and private repositories based upon industry-standard Git commands.
- **Actions:** Allows for the creation of automation workflows. These workflows can include environment variables and customized scripts.
- **Packages:** The majority of the world's open-source projects are already contained in GitHub repositories. GitHub makes it easy to integrate with this code and with other third-party offerings.
- **Security:** Provides detailed code scanning and review features, including automated code review assignment.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Explore an authorization and access strategy

Completed

- 3 minutes

Azure DevOps Services uses enterprise-grade authentication. To protect and secure your data, you can use:

- Microsoft account.
- GitHub account.
- Microsoft Entra ID.

Tools like Visual Studio and Azure DevOps natively support the use of Microsoft Accounts and Microsoft Entra ID. Eclipse can also support this form of authentication if you install a Team Explorer Everywhere plug-in.

Personal access tokens

Use personal access tokens (PAT) for tools that don't directly support Microsoft accounts or Microsoft Entra ID for authentication. You can use it if you want them to integrate with Azure DevOps.

For example, tools like:

- Git-based repositories.
- NuGet.

- Xcode.

These tokens can be set up using Git Credential managers, or you can create them manually.

Personal access tokens are also helpful when establishing access to command-line tools, external tools, and tasks in build pipelines.

Also, when calling REST-based APIs, you don't have a UI popping out to do the authentication. When access is no longer required, you can revoke the personal access token.

Security groups

Azure DevOps is pre-configured with default security groups.

Default permissions are assigned to the default security groups. You can also configure access at the organization, collection, and project or object levels.

In the organization settings in Azure DevOps, you can configure app access policies. Based on your security policies, you might allow alternate authentication methods, enable third-party applications to access via OAuth, or even allow anonymous access to some projects.

For even tighter control, you can use Conditional Access policies. These offer simple ways to help secure resources such as Azure DevOps when using Microsoft Entra ID for authentication.

Multifactor authentication

Conditional Access policies such as multifactor authentication can help to minimize the risk of compromised credentials.

As part of a Conditional Access policy, you might require:

- Security group membership.
- A location or network identity.
- A specific operating system.
- A managed device or other criteria.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Migrate or integrate existing work management tools

Completed

- 2 minutes

Both Azure DevOps and GitHub can be integrated with different kinds of work management tools.

As an example, in the Visual Studio Marketplace, Microsoft offers Trello integration tooling.

Migrating from other work management tools to Azure DevOps takes considerable planning.

Most work management tools are highly configurable by the end user. There might not be a tool available that will do the migration without further configuration.

Jira

Jira is a commonly used work management tool.

In the Visual Studio Marketplace, Solidify offers a tool for Jira to Azure DevOps migration. It migrates in two phases. Jira issues are exported to files, and then the files are imported to Azure DevOps.

If you decide to write the migration code yourself, the following blog post provides a sample code that might help you get started: Migrate your project from Jira to Azure DevOps .

Other applications

Third-party organizations offer commercial tooling to assist with migrating other work management tools like:

- Aha.
- BugZilla.
- ClearQuest.
- And others to Azure DevOps.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Migrate or integrate existing test management tools

Completed

- 1 minute

Azure Test Plans track manual testing for sprints and milestones, allowing you to follow when that testing is complete.

Azure DevOps also has a Test Feedback extension available in the Visual Studio Marketplace. The extension is used to help teams do exploratory testing and provide feedback.

All team members and other stakeholders can use the extension to submit bugs or provide feedback. For example:

- Developers.
- Product owners.
- Managers.
- UX.
- UI engineers.
- Marketing teams.
- Early adopters.

For load tests, you can use Azure Load Testing. For more information, see What is Azure Load Testing? .

Other helpful testing tools:

Apache JMeter is open-source software written in Java and designed to load test, and measure performance.

Pester is a tool that can automate the testing of PowerShell code.

SoapUI is another testing framework for SOAP and REST testing.

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

For more information, search for Test Management at Visual Studio Marketplace.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Design a license management strategy

Completed

- 1 minute

When designing a license management strategy, you first need to understand your progress in the DevOps implementation phase.

If you have a draft of the architecture, you're planning for the DevOps implementation; you already know part of the resources to consume.

For example, you started with a version control-implementing Git and created some pipelines to build and release your code.

If you have multiple teams building their solutions, you don't want to wait in the queue to start building yours.

Probably, you want to pay for parallel jobs and make your builds run in parallel without depending on the queue availability.

To consider:

- What phase are you in?
- How many people are using the feature?
- How long are you willing to wait if in the queue for pipelines? Is this urgent? Is this a validation only?
- Should all users access all features? Are they Stakeholders? Basic users? Do they already have a Visual Studio license?
- Do you have an advanced Package Management strategy? Maybe you need more space for Artifacts.

For the latest, most up-to-date Azure DevOps pricing information, visit Azure DevOps Pricing .

For the latest, most up-to-date GitHub pricing information, visit GitHub Pricing .

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Knowledge check

Completed

- 4 minutes

Choose the best response for each question. Then select **Check your answers** .

Check your knowledge

1.

Which of the following choices isn't an Azure DevOps service/feature?

Azure Boards.

Azure Monitor.

Azure Repos.

2.

Which of the following is a correct statement about Azure DevOps?

Azure DevOps only works in on-premises environments. If you want to use a cloud service, you need to choose GitHub.

Azure DevOps and GitHub are only available in the cloud.

Azure DevOps and GitHub provide options to work on-premises and in the cloud.

3.

Which of the following choices does Azure DevOps Services enterprise-grade authentication support?

Microsoft Account, Microsoft Entra ID, GitHub Account.

Microsoft Account, Microsoft Entra ID.

GitHub Account.

Check your answers

You must answer all questions before checking your work.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Summary

Completed

- 1 minute

This module explored the basics of Azure DevOps and GitHub and how to start using them to implement DevOps.

You learned how to describe the benefits and usage of:

- Design a tool integration strategy.
- Design a license management strategy (for example, Azure DevOps and GitHub users).
- Design a strategy for end-to-end traceability from work items to working software.
- Design an authentication and access strategy.
- Design a strategy for integrating on-premises and cloud resources.

Learn more

- Azure DevOps Services | Microsoft Azure .
- Features | GitHub .
- Pricing Calculator | Microsoft Azure .
- Azure DevOps Server to Services Migration overview - Azure DevOps | Microsoft Docs .
- Azure DevOps Services Pricing | Microsoft Azure .
- GitHub Pricing .

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Plan Agile with GitHub Projects and Azure Boards

Introduction

Completed

- 3 minutes

This module introduces you to GitHub Projects, GitHub Project Boards, and Azure Boards. It explores ways to link Azure Boards and GitHub, configure GitHub Projects and Project views, and manage work with GitHub Projects.

Learning objectives

After completing this module, students and professionals can:

- Describe GitHub Projects and Azure Boards.
- Link Azure Boards and GitHub.
- Configure and Manage GitHub Projects and boards.
- Customize Project views.

Prerequisites

- Understanding of what DevOps is and its concepts.
- Familiarity with version control principles is helpful but isn't necessary.
- Beneficial to have experience in an organization that delivers software.
- You need to create a GitHub account at GitHub.com and a project for some exercises. If you don't have it yet, see: Join GitHub · GitHub . If you already have your GitHub account, create a new repository Creating a new repository - GitHub Docs .

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Introduction to GitHub Projects and Project boards

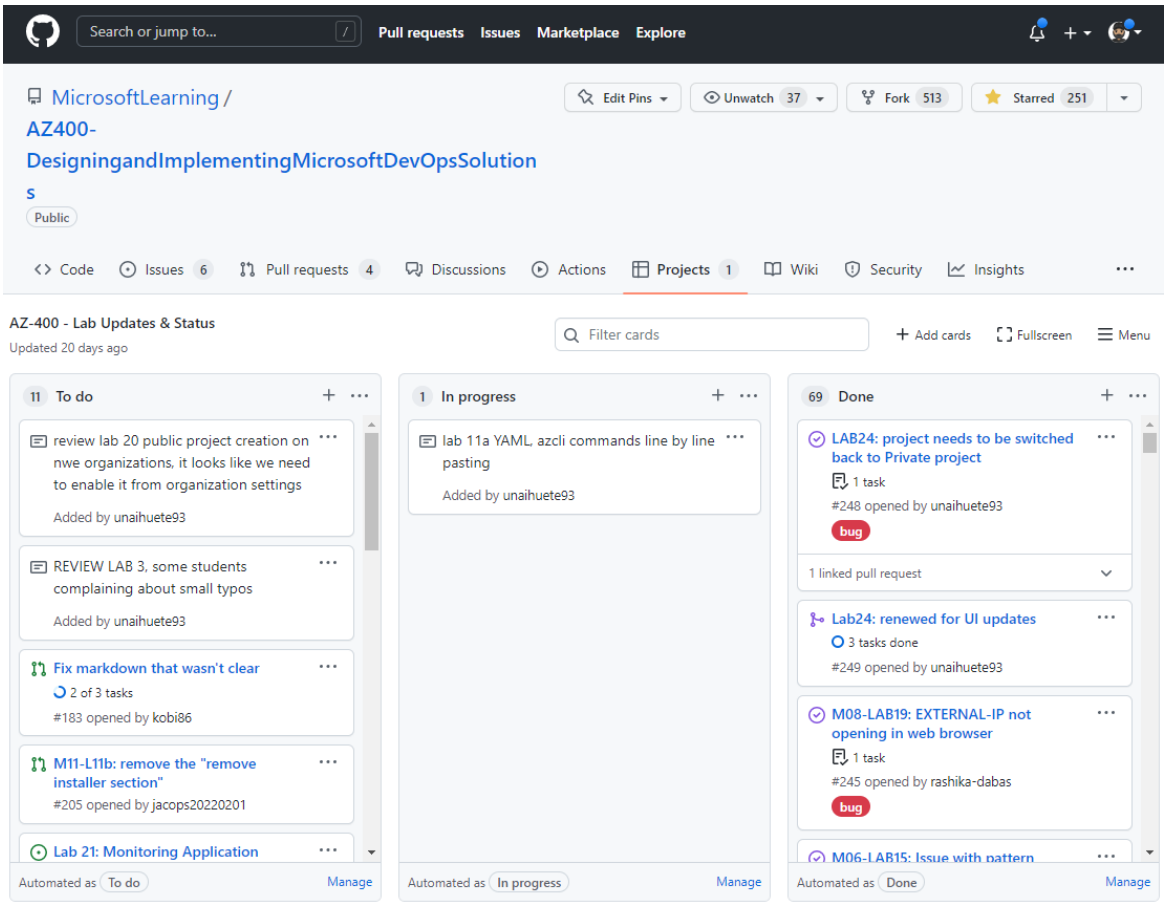
Completed

- 3 minutes

Project Boards

During the application or project lifecycle, it's crucial to plan and prioritize work. With Project boards, you can control specific feature work, roadmaps, release plans, etc.

Project boards are made up of issues, pull requests, and notes categorized as cards you can drag and drop into your chosen columns. The cards contain relevant metadata for issues and pull requests, like labels, assignees, the status, and who opened it.



There are different types of project boards:

- **User-owned project boards:** Can contain issues and pull requests from any personal repository.
- **Organization-wide project boards:** Can contain issues and pull requests from any repository that belongs to an organization.
- **Repository project boards:** Are scoped to issues and pull requests within a single repository.

To create a project board for your organization, you must be an organization member.

It's possible to use templates to set up a new project board that will include columns and cards with tips. The templates can be automated and already configured.

Templates	Description

Templates	Description
Basic kanban	Track your tasks with: To do, In progress, and Done columns.
Automated kanban	Cards automatically move between: To do, In progress, and Done columns.
Automated kanban with review	Cards automatically move between: To do, In progress, and Done columns, with extra triggers for pull request review status.
Bug triage	Triage and prioritize bugs with: To do, High priority, Low priority, and Closed columns.

For more information about Project boards, see:

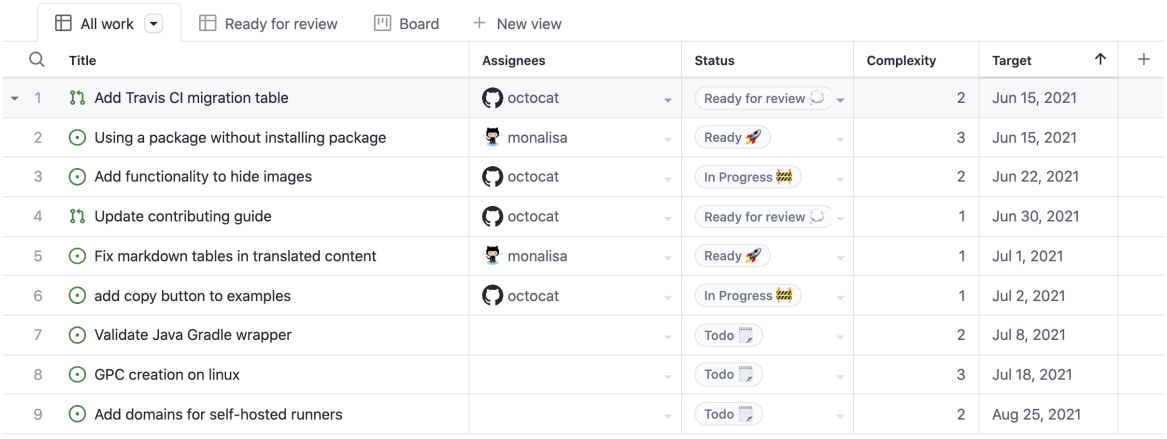
- Creating a project board .
- Editing a project board .
- Copying a project board .
- Adding issues and pull requests to a project board .
- Project board permissions for an organization .

Projects

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

A project is a customizable spreadsheet in which you can configure the layout by filtering, sorting, grouping your issues and PRs, and adding custom fields to track metadata.

You can use different views such as Board or spreadsheet/table.



If you change your pull request or issue, your project reflects that change.

You can use custom fields in your tasks. For example:

- A date field to track target ship dates.
- A number field to track the complexity of a task.

- A single select field to track whether a task is Low, Medium, or High priority.
- A text field to add a quick note.
- An iteration field to plan work week-by-week, including support for breaks.

For more information about Projects, see:

- Creating a project .
- Managing iterations in projects .
- Customizing your project views .
- Automating projects .

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Introduction to Azure Boards

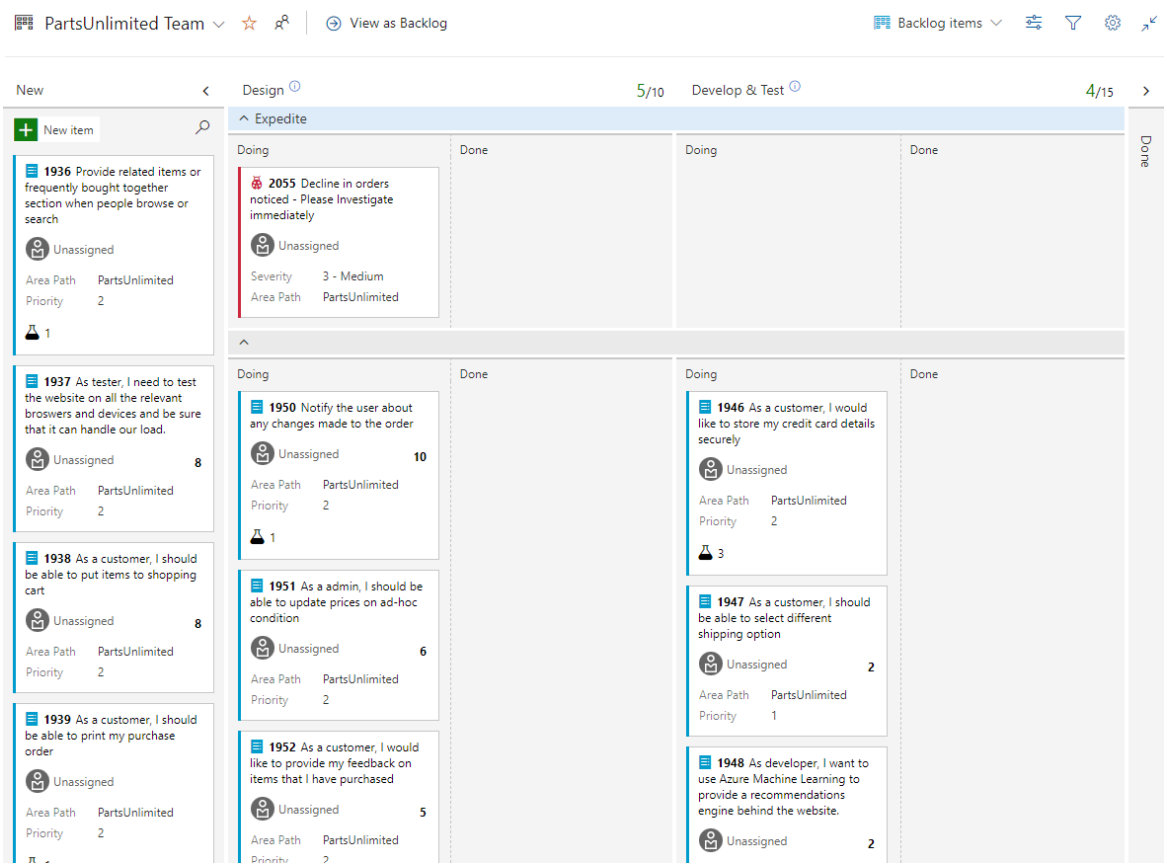
Completed

- 3 minutes

Azure Boards is a customizable tool to manage software projects supporting Agile, Scrum, and Kanban processes by default. Track work, issues, and code defects associated with your project. Also, you can create your custom process templates and use them to create a better and more customized experience for your company.

You have multiple features and configurations to support your teams, such as calendar views, configurable dashboards, and integrated reporting.

The Kanban board is one of several tools that allows you to add, update, and filter user stories, bugs, features, and epics.



You can track your work using the default work item types such as user stories, bugs, features, and epics. It's possible to customize these types or create your own. Each work item provides a standard set of system fields and controls, including Discussion for adding and tracking comments, History, Links, and Attachments.

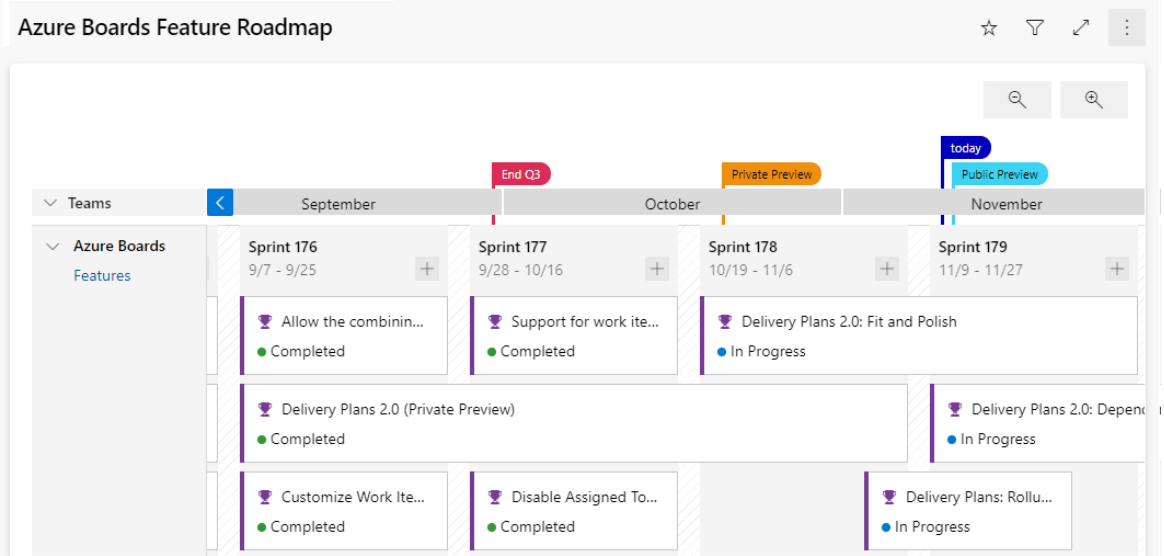
If you need to create reports or a list of work with specific filters, you can use the queries hub to generate custom lists of work items.

Queries support the following tasks:

- Find groups of work items with something in common.
- Triage work to assign to a team member or sprint and set priorities.
- Perform bulk updates.
- View dependencies or relationships between work items.
- Create status and trend charts that you can optionally add to dashboards.

Delivery plans

It's possible to create another view with deliverables and track dependencies across several teams in a calendar view using Delivery Plans.



Delivery plans are fully interactive, supporting the following tasks:

- View up to 15 team backlogs, including a mix of backlogs and teams from different projects.
- View custom portfolio backlogs and epics.
- View work that spans several iterations.
- Add backlog items from a plan.
- View rollup progress of features, epics, and other portfolio items.
- View dependencies that exist between work items.

For more information about Azure Boards, see:

- Azure Boards documentation | Microsoft Docs .
- Reasons to start using Azure Boards | Microsoft Docs .
- GitHub and Azure Boards .

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Link GitHub to Azure Boards

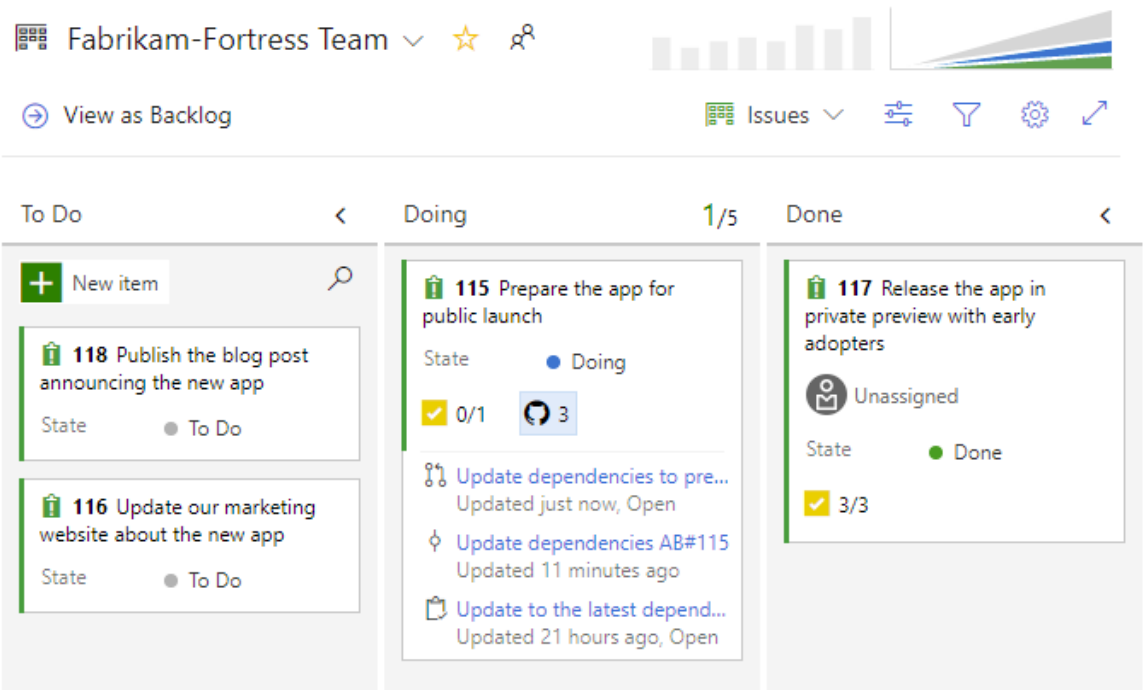
Completed

- 2 minutes

Use GitHub, track work in Azure Boards

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

Connect Azure Boards with GitHub repositories, enabling linking GitHub commits, pull requests, and issues to work items in Boards.

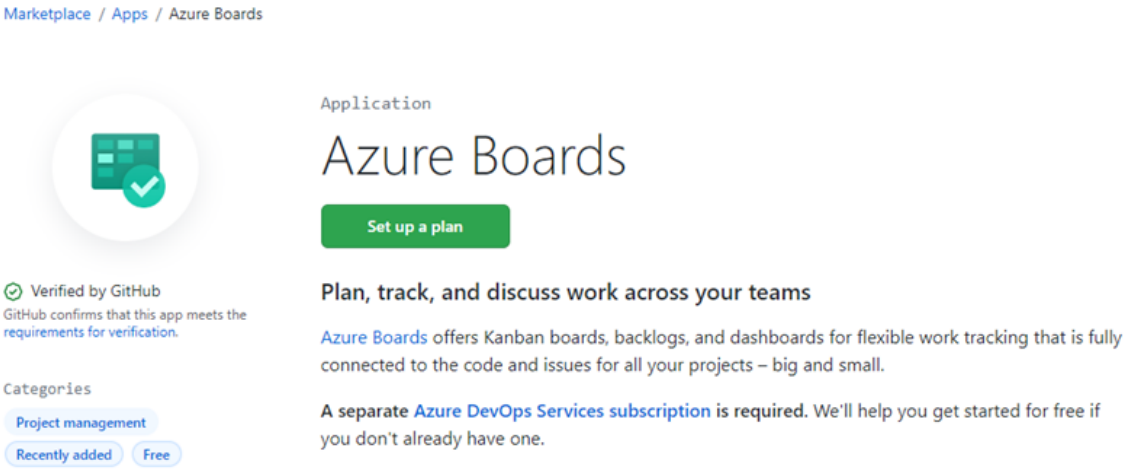


Azure Boards App

The integration is created using the Azure Boards App, acting as a bridge between Azure Boards and GitHub.

To install the app, you must be an administrator or owner of the GitHub repository or the GitHub organization.

The app is installed from the GitHub Marketplace. [Azure Boards App]



Authenticating to GitHub

Azure Boards can connect to GitHub. For GitHub in the cloud, when adding a GitHub connection, the authentication options are:

- Username/Password
- Personal Access Token (PAT)

For a walkthrough on making the connection, see: Connect Azure Boards to GitHub .

You can configure other Azure Boards/Azure DevOps Projects, GitHub.com repositories, or change the current configuration from the Azure Boards app page.

Once you've integrated Azure Boards with GitHub using the Azure Boards app, you can add or remove repositories from the web portal for Azure Boards.

Supported integration scenarios

Azure Boards-GitHub integration supports the following connections:

- From GitHub:
    - Support integration for all repositories for a GitHub account or organization or select repositories.
    - Add or remove GitHub repositories participating in the integration and configure the project they connect to.
    - Suspend Azure Boards-GitHub integration or uninstall the app.
- From Azure Boards:
    - Connect one or more GitHub repositories to an Azure Boards project.
    - Add or remove GitHub repositories from a GitHub connection within an Azure Boards project.
    - Completely remove a GitHub connection for a project.
    - Allow a GitHub repository to connect to one or more Azure Boards projects within the same Azure DevOps organization or collection.

Azure Boards-GitHub integration supports the following operational tasks:

- Create links between work items and GitHub commits, pull requests, and issues based on GitHub mentions.
- Support state transition of work items to a Done or Completed state when using GitHub mention by using fix, fixes, or fixed.
- Support full traceability by posting a discussion comment to GitHub when linking from a work item to a GitHub commit, pull request, or issue.
- Show linked to GitHub code artifacts within the work item Development section.
- Show linked to GitHub artifacts as annotations on Kanban board cards.
- Support status badges of Kanban board columns added to GitHub repositories.

The following tasks aren't supported at this time:

- Query for work items with links to GitHub artifacts. However, you can query for work items with an External Link Count greater than 0.

Note

Reference: Azure Boards-GitHub integration .

For more information, see:

- Change GitHub repository access, or suspend or uninstall the integration .
- Add or remove GitHub repositories .
- Link GitHub commits, pull requests, and issues to work items for details on linking to work items .

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

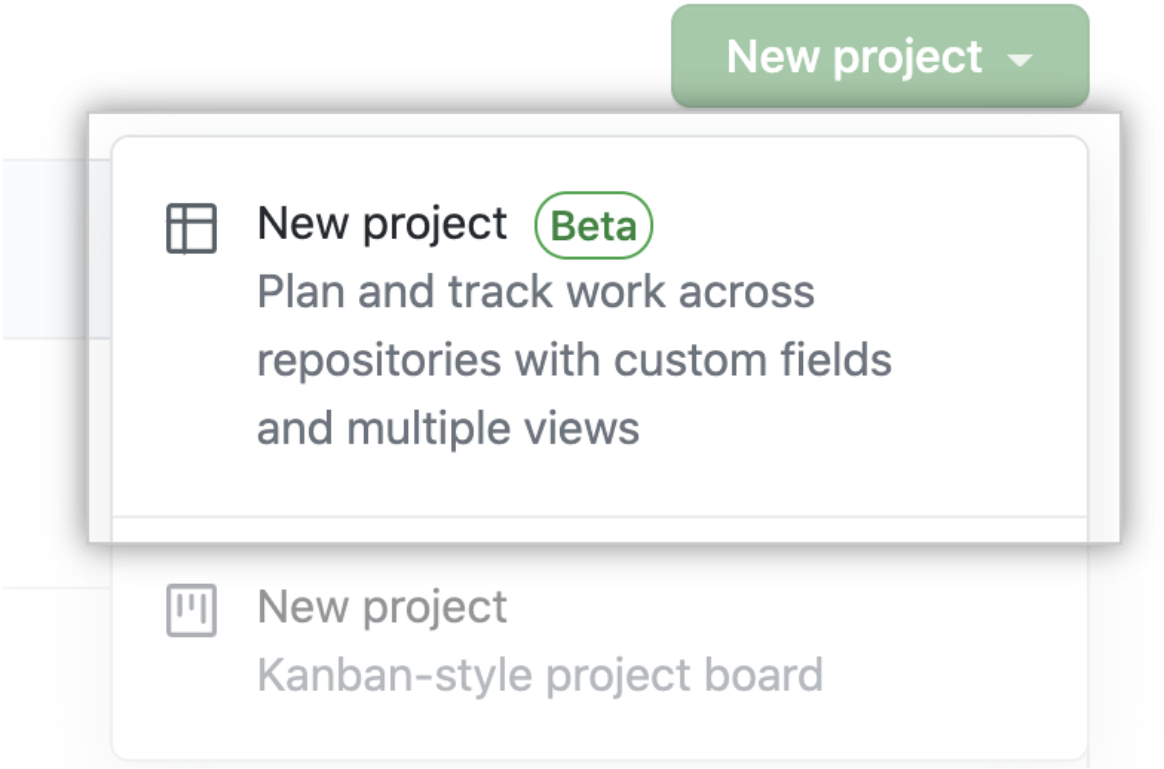
Configure GitHub Projects

Completed

- 3 minutes

Create a project

To start working with GitHub Projects, you first need to create an organization or user project.



To create an **organization project** :

1. On GitHub, navigate to the main page of your organization.
2. Click **Projects** .
3. Select the **New project** drop-down menu and click **New project.**

To create a **user project** :

1. On any GitHub page, click on your avatar, then select **Your projects** .
2. Select the **New project** drop-down menu and click **New project** .

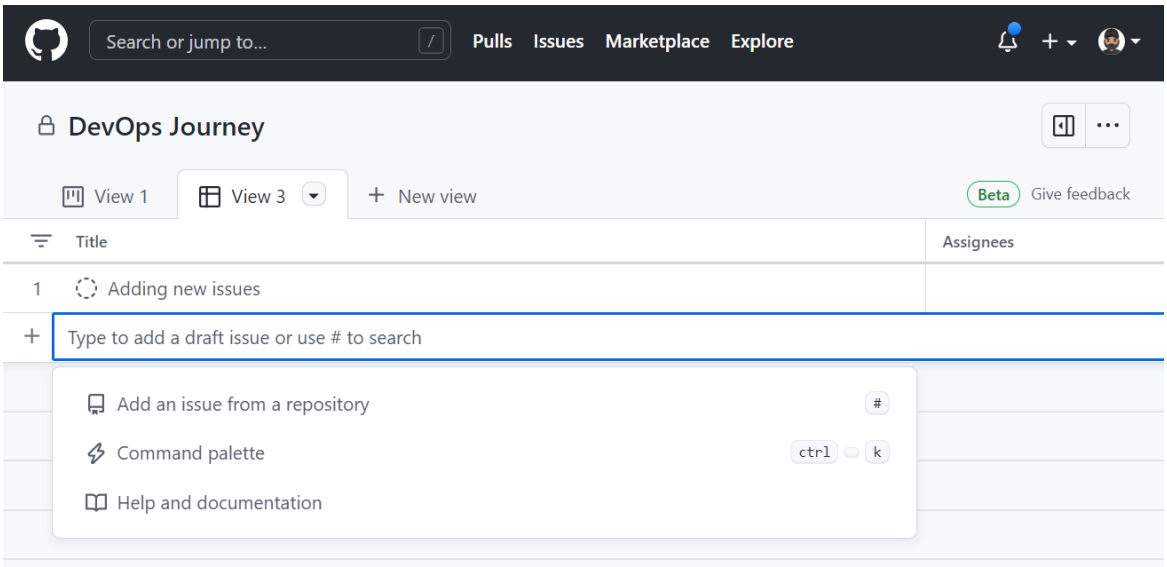
Create a project **description** or a **README** file:

1. Navigate to your project.
2. In the top-right, click to open the menu.
3. In the menu, click **Settings** .
4. Under Add a description, type your description in the text box and click **Save** .
5. To update your project's README, type your content in the text box under README.
6. Click Save.

Adding issues

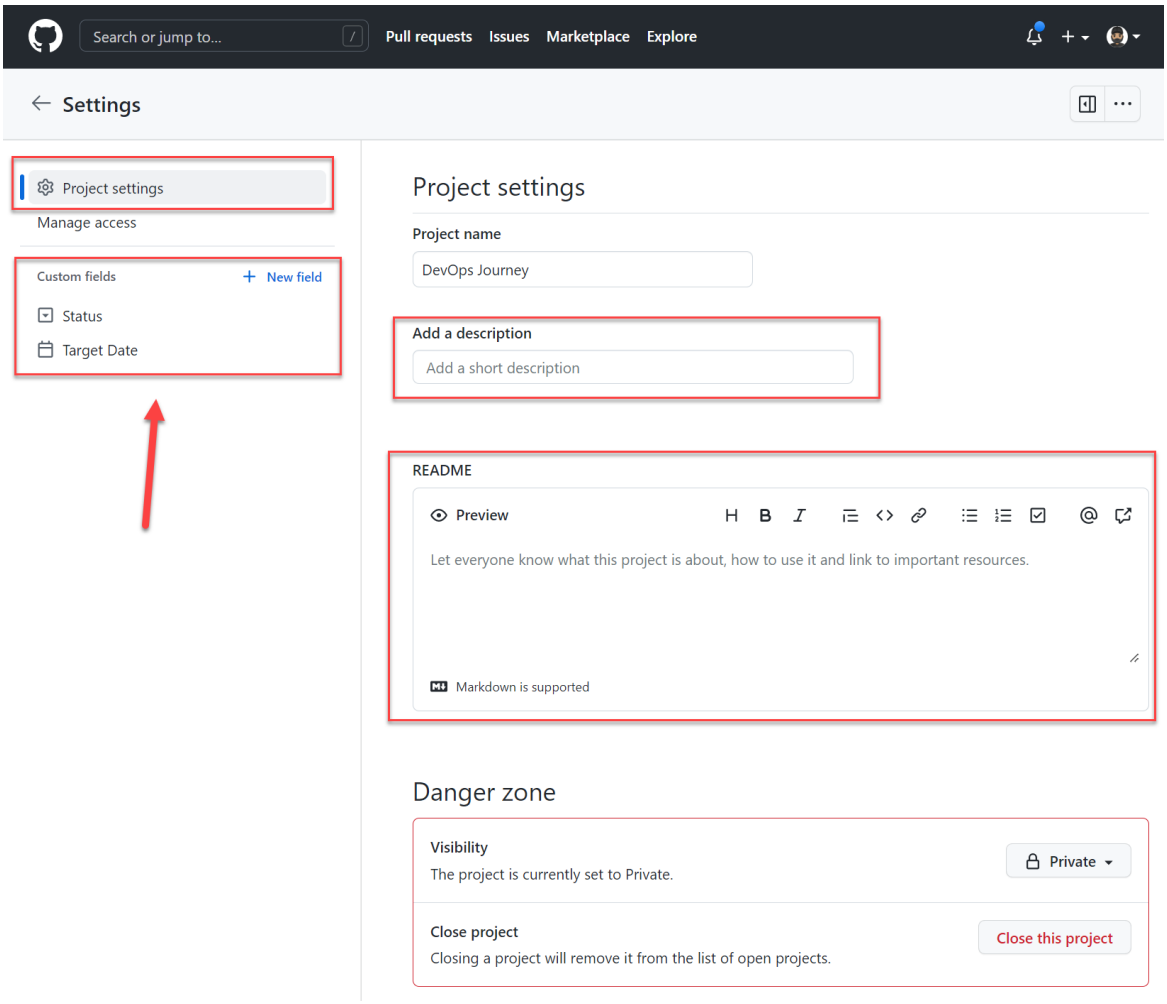
When your new project initializes, it prompts you to add items.

Click on the plus (+) sign to add more issues.

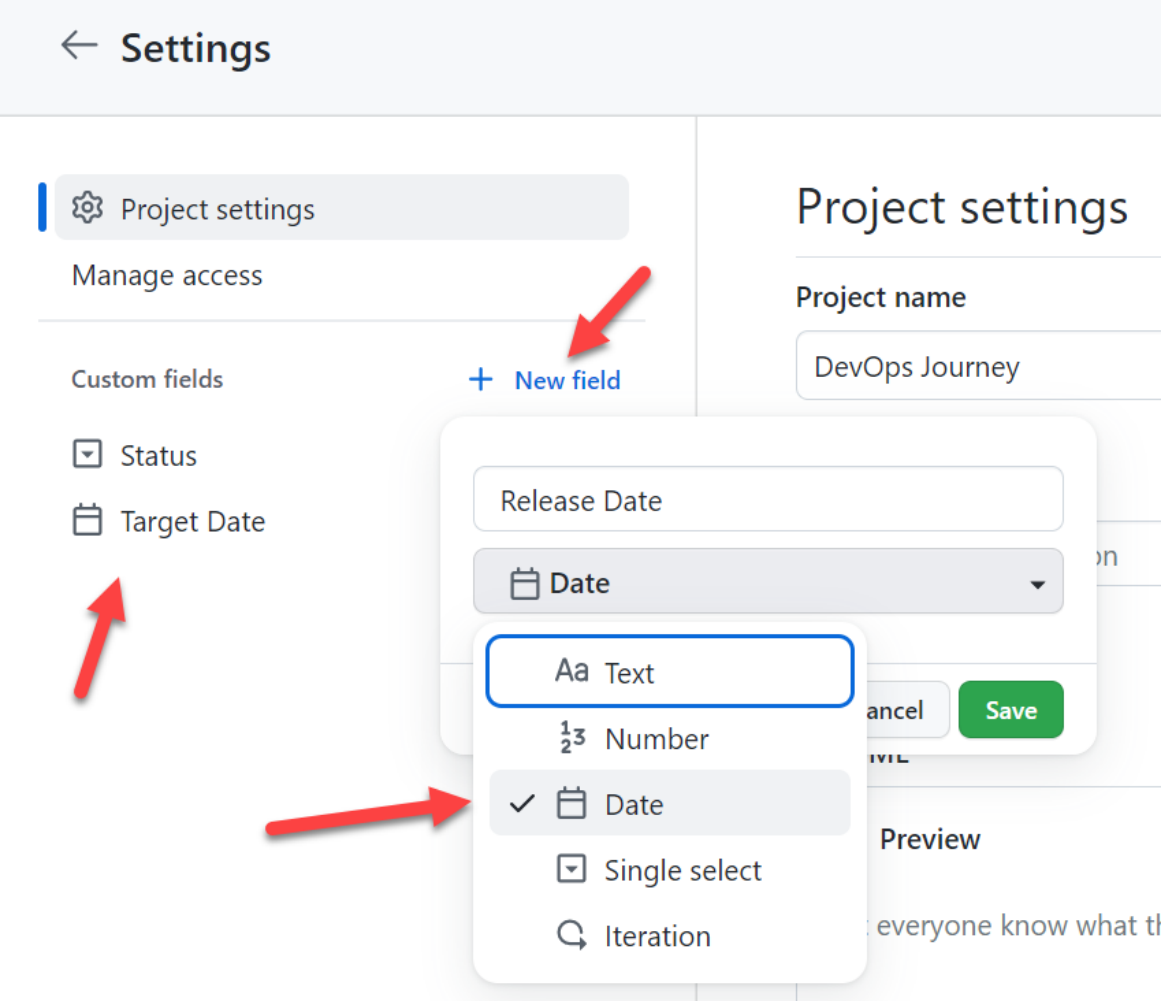


Project settings and permissions

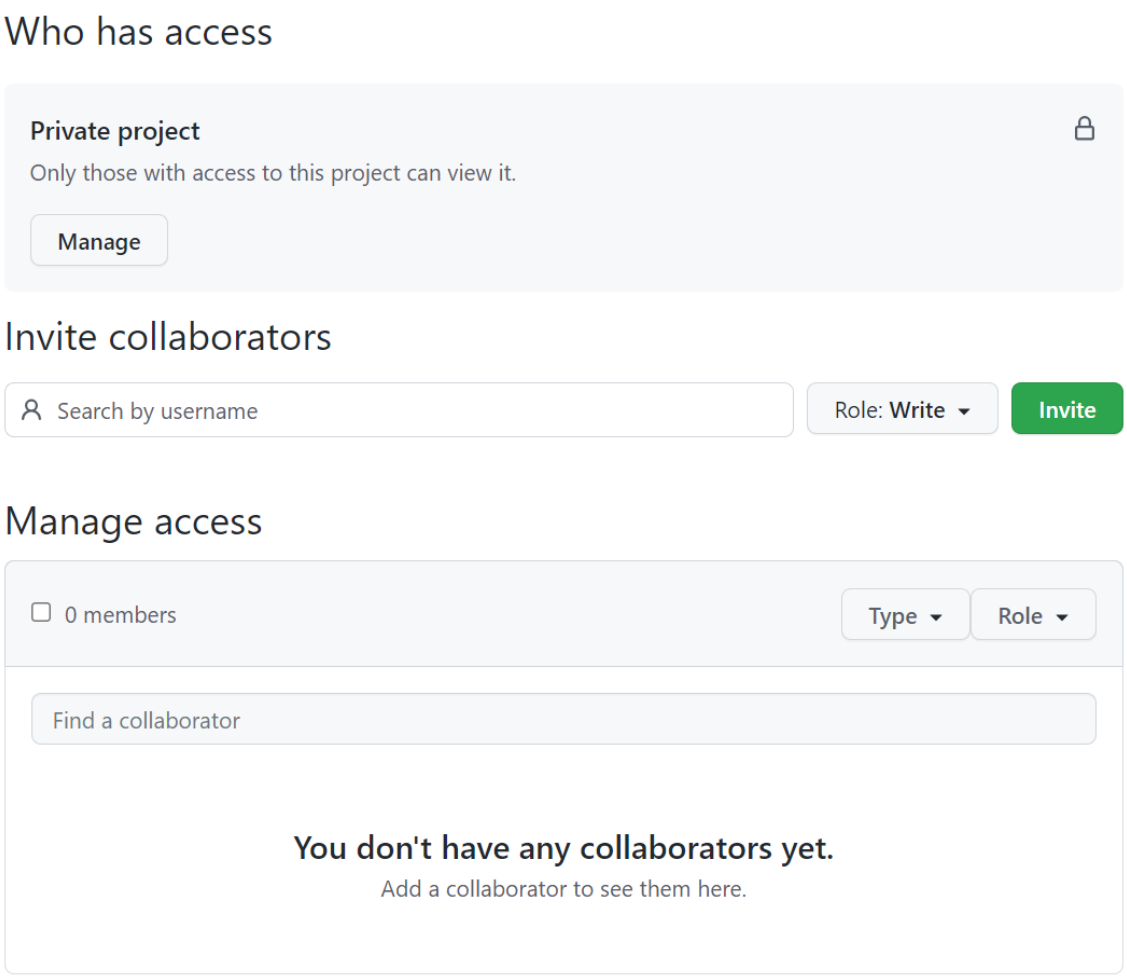
You can view and make quick changes to your project description and README by navigating to your project and clicking on the top right.



You can create custom fields to add to your project.



Manage access and permissions by adding collaborators.



For more information about Projects, see:

- Quickstart for projects - GitHub Docs .
- Creating an issue - GitHub Docs .

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Manage work with GitHub Project boards

Completed

- 2 minutes

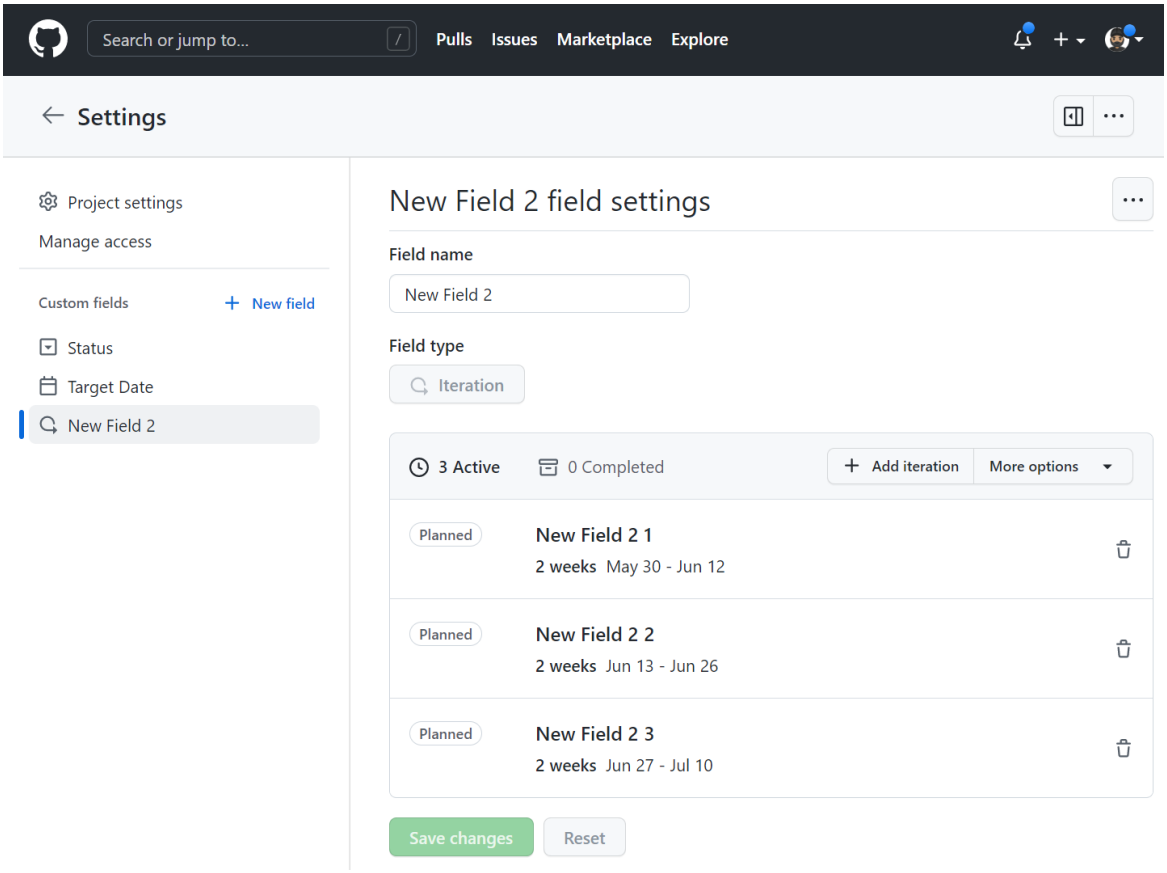
GitHub Projects allow you to control project deliverables, release dates, and iterations to plan upcoming work.

You can create an iteration to:

- Associate items with specific repeating blocks of time.
- Set to any length of time.
- Include breaks.

It's possible to configure your project to group by iteration to visualize the balance of upcoming work.

When you first create an iteration field, three iterations are automatically created. You can add other iterations if needed.



Iteration field

You can use the command palette or the project's interface to create an iteration field.

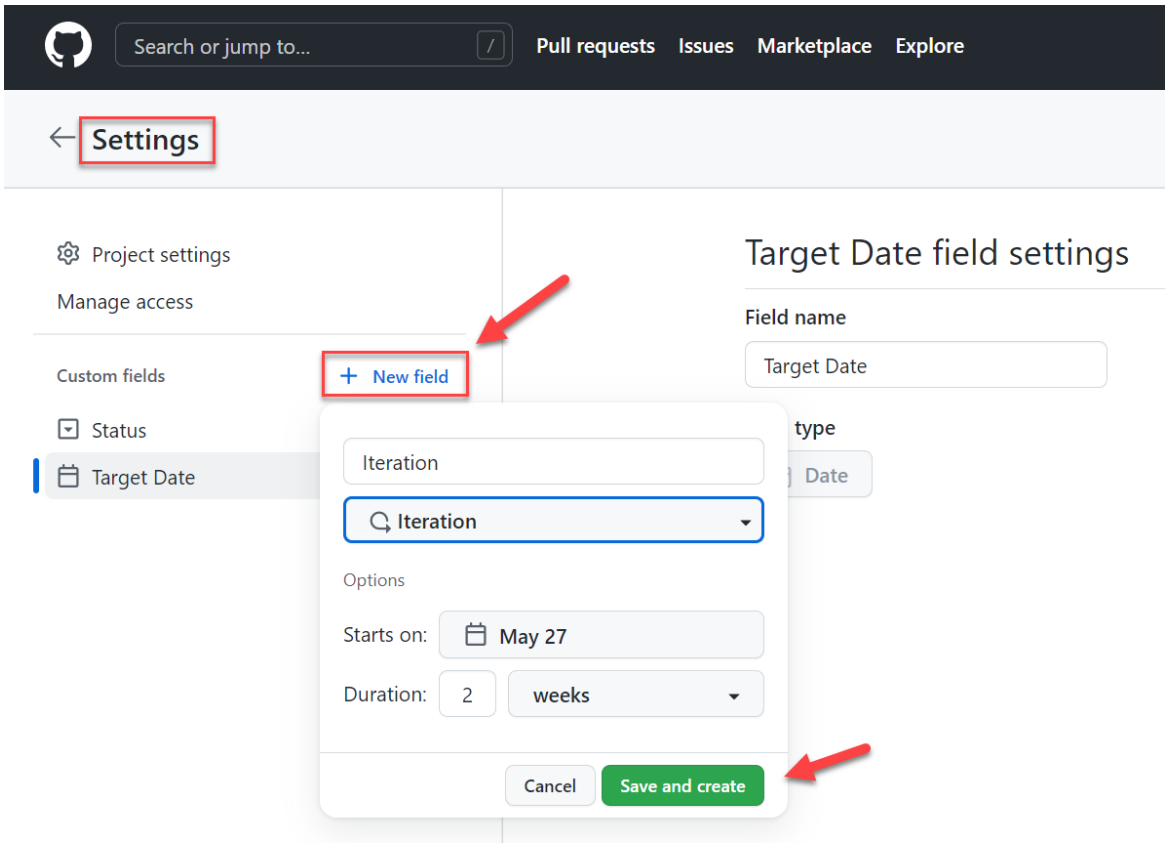
Tip

To open the project command palette, press Ctrl+K (Windows/Linux) or Command+K (Mac).

Start typing any part of "Create new field". When "Create new field" displays in the command palette, select it.

Or follow the steps using the interface:

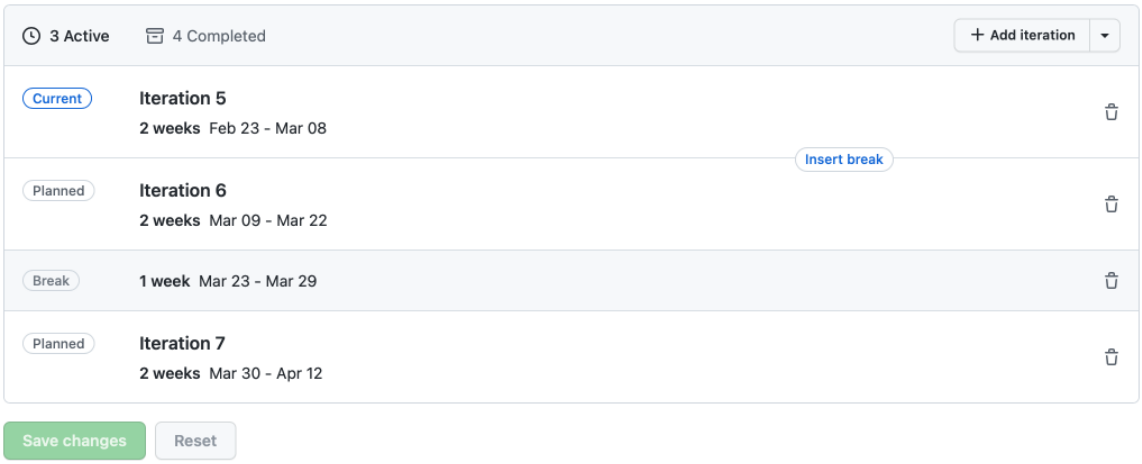
1. Navigate to your project.
2. Click in the plus (+) sign in the rightmost field header. A drop-down menu with the project fields will appear.
3. Click in the **New field** .
4. Enter a name for the new iteration field.
5. Select the dropdown menu below and click Iteration.
6. (Optional) Change the starting date from the current day, select the calendar dropdown next to Starts on, and click on a new starting date.
7. To change the duration of each iteration, type a new number, then select the dropdown and click either days or weeks.
8. Click Save and create.



Adding new iterations

1. Navigate to your project.

2. In the top-right, click to open the menu.
3. In the menu, click **Settings** to access the project settings.
4. Click the name of the iteration field you want to adjust.
5. To add a new iteration of the same duration, click **Add iteration** .
6. (Optional) Customize the duration of the new iteration and when it starts.
    1. Click next to Add iteration.
    2. Select a starting date and duration.
    3. Click **Add** .
7. Click Save changes.



Also, you can insert breaks into your iterations to communicate when you're taking time away from scheduled work.

For more information about iterations, see:

- Managing iterations in projects (beta) - GitHub Docs .
- Best practices for managing projects (beta) - GitHub Docs .

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

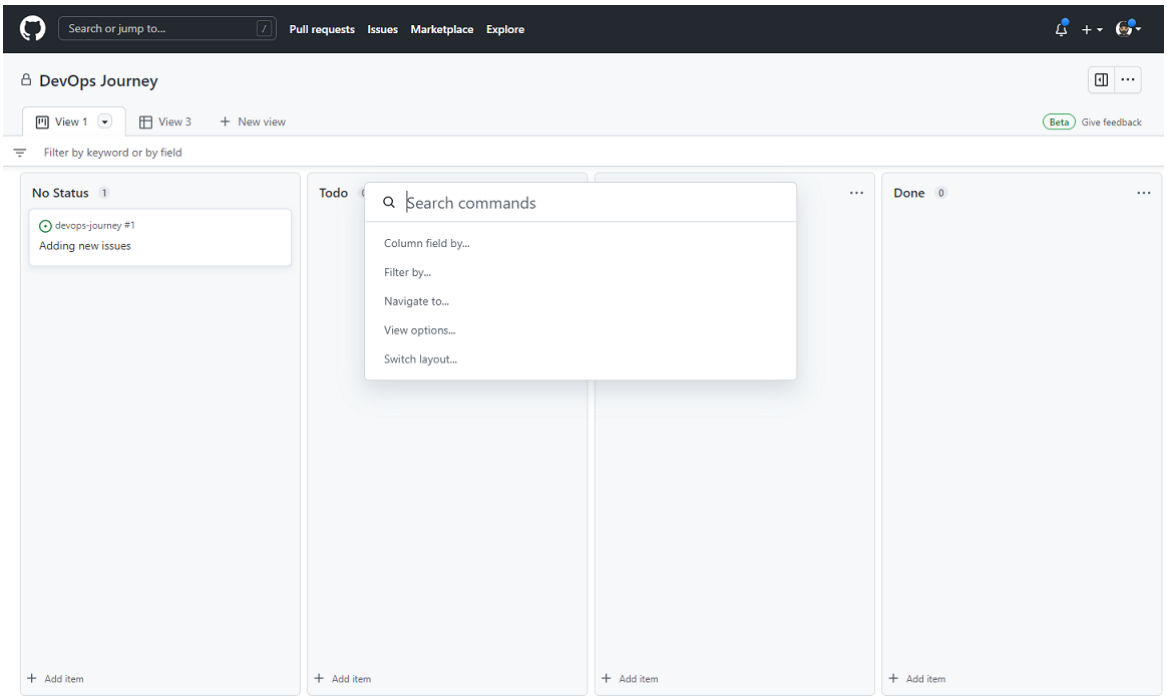
Customize Project views

Completed

- 3 minutes

Using Projects views, you can organize information by changing the layout, grouping, sorting, and filtering your work.

You can create and use different visualizations, for example, Board view:



Project command palette

Use the project command palette to change settings and run commands in your project.

1. Open the project command palette by pressing **Command + K (Mac)** or **Ctrl + K (Windows/Linux)** .
2. Type any command part or navigate through the command palette window to find a command.

You have multiple commands to apply, such as:

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

Note

For more information about GitHub Command Palette, see GitHub Command Palette - GitHub Docs .

Also, you can perform changes using the interface.

Creating a project view

Project views allow you to view specific aspects of your project. Each view is displayed on a separate tab in your project.

For example, you can have:

- A view that shows all items not yet started (filter on Status).
- A view that shows the workload for each team (group by a custom Team field).
- A view that shows the items with the earliest target ship date (sort by a date field).

To add a new view:

1. To open the project command palette, press **Command + K (Mac)** or **Ctrl + K (Windows/Linux)** .
2. Start typing **New view** (to create a new view) or **Duplicate view** (to duplicate the current view).
3. Choose the required command.
4. The new view is automatically saved.

For more information about projects (beta), see:

- About projects (beta) .
- Creating a project (beta) .

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

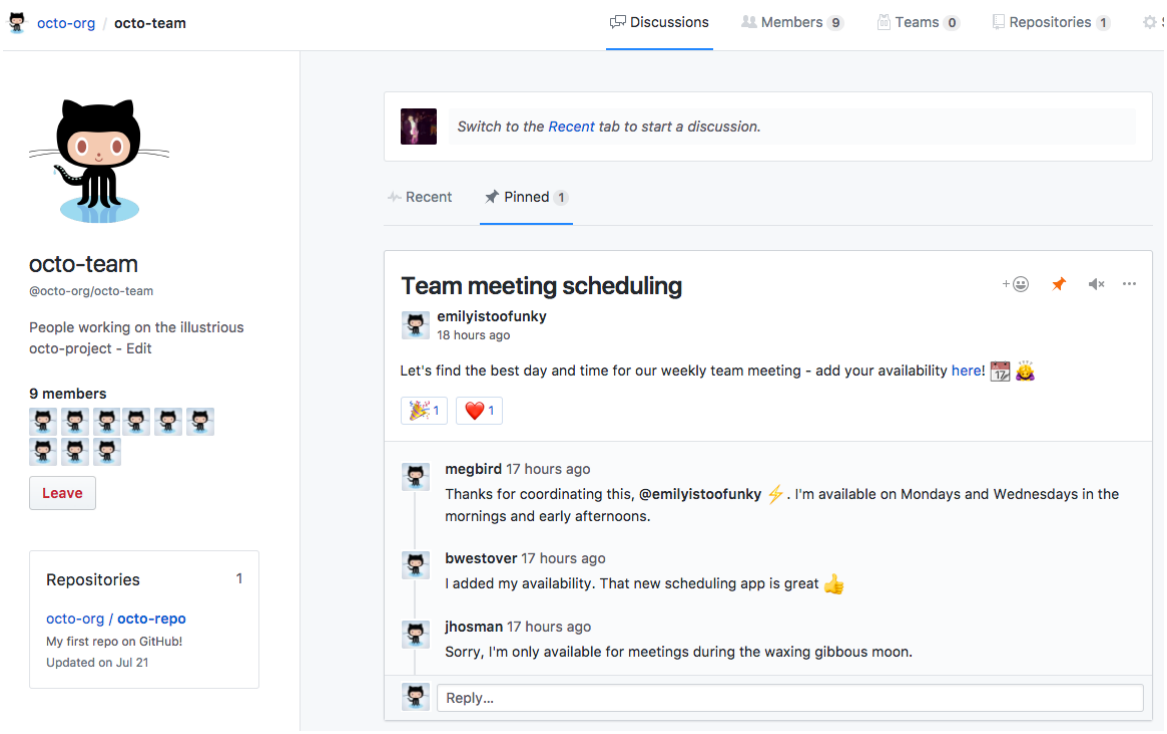
Collaborate using team discussions

Completed

- 3 minutes

GitHub discussions can help make your team plan together, update one another, or talk about any topic you'd like in discussion posts on your team's page in an organization.

You can use team discussions for conversations that span across projects or repositories (issues, pull requests, etc.). Instead of opening an issue in a repository to discuss an idea, you can include the entire team by having a conversation in a team discussion.



With team discussions, you can:

- Post on your team's page or participate in a public discussion.
- Link to any team discussion to reference it elsewhere.
- Pin important posts to your team's page.
- Receive email or web notifications.

Team discussions are available in organizations by default.

You can also use organization discussions to facilitate conversations across your organization.

For more information about team discussion, see:

- Enabling or disabling GitHub Discussions for an organization .
- Quickstart for communicating on GitHub .
- About teams .
- Creating a team discussion .
- Editing or deleting a team discussion .

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Agile Plan and Portfolio Management with Azure Boards

Completed

- 57 minutes

Estimated time: 60 minutes.

Lab files: none.

Scenario

In this lab, you'll 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. You'll explore the product backlog, sprint backlog, and task boards that can track the flow of work during an iteration. We'll also look at the enhanced tools in this release to scale for larger teams and organizations.

Objectives

After completing this lab, you'll be able to:

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

Requirements

- This lab requires **Microsoft Edge** or an Azure DevOps-supported browser .
- **Set up an Azure DevOps organization:** If you don't already have an Azure DevOps organization that you can use for this lab, create one by following the instructions available at Create an organization or project collection .

Exercises

During this lab, you'll complete the following exercises:

- Exercise 0: Configure the lab prerequisites.
- Exercise 1: Manage an Agile project.
- Exercise 2 (optional): Define dashboards.

Launch Exercise

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Knowledge check

Completed

- 3 minutes

Choose the best response for each question. Then select **Check your answers** .

Check your knowledge

1.

Which of the following choices best describes the Azure DevOps and GitHub integration?

○

Azure Boards have direct integration with Azure Repos, but it can also be integrated with GitHub to plan and track work linking commits, PRs, and issues.

○

Azure Boards has direct integration with Azure Repos. Azure Repos only integrates with GitHub using extensions from Marketplace for the read-only track.

○

Azure Boards has direct integration with GitHub for tracking activities and moving tasks on both sides Azure Boards to GitHub and GitHub to Azure Repos.

2.

Which of the following Project boards types can contain issues and pull requests from any personal repository?

○

User-owned project boards.

○

Organization-wide project boards.

○

Repository project boards

3.

Which of the following choices isn't a Project Boards supported template by default?

Basic kanban.

Automated kanban with review.

Automated CMMI.

Check your answers

You must answer all questions before checking your work.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Summary

Completed

- 3 minutes

This module introduced you to GitHub Projects, GitHub Project Boards, and Azure Boards. It explored ways to link Azure Boards and GitHub, configure GitHub Projects and Project views, and manage work with GitHub Projects.

You learned how to describe the benefits and usage of:

- Describe GitHub Projects and Azure Boards.
- Link Azure Boards and GitHub.
- Configure and Manage GitHub Projects and boards.
- Customize Project views.

Learn more

- Quickstart for projects (beta) - GitHub Docs .
- About project boards - GitHub Docs .
- What is Azure Boards? Tools to manage software development projects. - Azure Boards | Microsoft Docs .

- Azure Boards-GitHub integration - Azure Boards | Microsoft Docs .
- Managing iterations in projects (beta) - GitHub Docs .
- Quickstart for projects (beta) - GitHub Docs .
- Best practices for managing projects (beta) - GitHub Docs .
- Customizing your project (beta) views - GitHub Docs .
- About team discussions - GitHub Docs .

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Introduction to source control

Introduction

Completed

- 1 minute

You can think of source control as an essential everyday practice. Versioning is a standard part of the developer's routine and, if used correctly, can save organizations enormous costs and resources.

The source control is a common-sense aspect of programming. It is essential from time to time to look at why we do what we do and how versioning impacts the entire value stream at an organization.

This module introduces you to the basics of source control, exploring benefits and best practices.

Learning objectives

After completing this module, students and professionals can:

- Understand source control.
- Apply best practices for source control.
- Describe the benefits of using source control.

Prerequisites

- Understanding of what DevOps is and its concepts.
- Familiarity with version control principles is helpful but is not necessary.
- Beneficial to have experience in an organization that delivers software.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Explore DevOps foundational practices

Completed

- 3 minutes

The State of DevOps Report 2021 highlights version control in almost all stages of DevOps evolution.

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.**

It is helpful for non-developers in an organization to understand the fundamentals of the discipline.

Also, it is so deeply rooted in the daily life of software engineers. Essential if those individuals decide which version control tools and platforms to use.

Version control is essential for all software development projects and is vital at large businesses and enterprises.

Enterprises have many stakeholders. For example:

- Distributed teams.
- Strict processes and workflows.
- Siloed organizations.
- Hierarchical organizations.

All those characteristics represent coordination and integration challenges when it comes to merging and deploying code.

Companies within highly regulated industries need a practical way to ensure that all standards are met appropriately and mitigate risk—for example, banking and healthcare.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

What is source control?

Completed

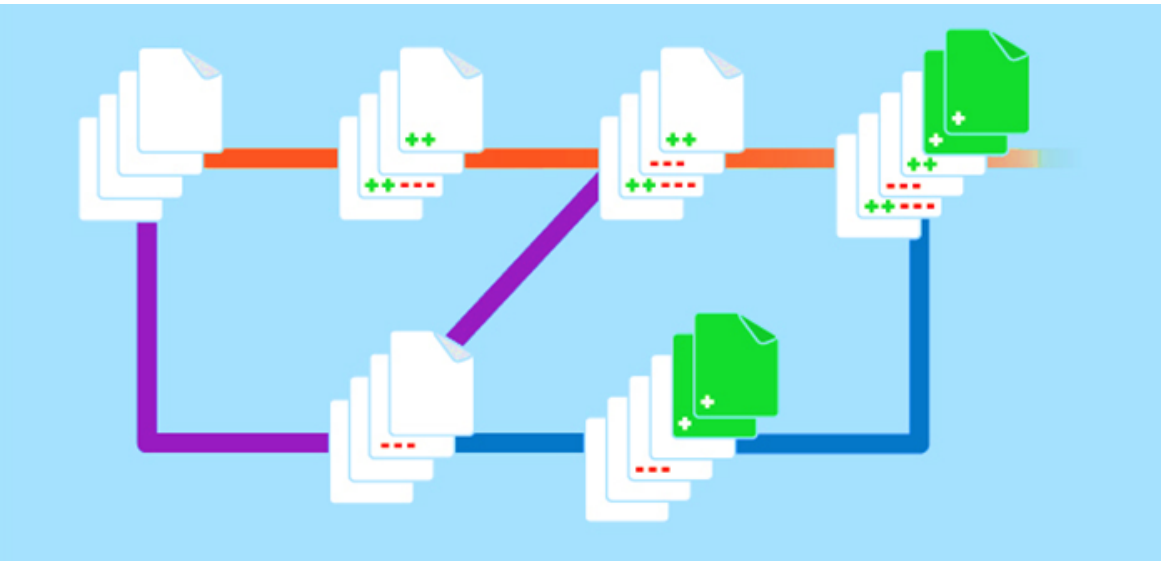
- 2 minutes

A Source control system (or version 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. Source control is an essential tool for multi-developer projects.

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. Also, it helps to resolve conflicts when merging contributions from multiple sources.

For most software teams, the source code is a repository of invaluable knowledge and understanding about the problem domain that the developers have collected and refined through careful effort.

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



Without version control, you're tempted to keep multiple copies of code on your computer. It could be dangerous. Easy to change or delete a file in the wrong code copy, potentially losing work.

Version control systems solve this problem by managing all versions of your code but presenting you with a single version at a time.

Tools and processes alone aren't enough to accomplish the above, such as adopting Agile, Continuous Integration, and DevOps. Believe it or not, all rely on a solid version control practice.

Version control is about keeping track of every change to software assets—tracking and managing the who, what, and when. Version control is the first step needed to assure quality at the source, ensure flow and pull value, and focus on the process. All of these create value not just for the software teams but ultimately for the customer.

Version control is a solution for managing and saving changes made to any manually created assets. If changes are made to the source code, you can go back in time and easily roll back to previous-working versions.

Version control tools will enable you to see who made changes, when, and what exactly was changed.

Version control also makes experimenting easy and, most importantly, makes collaboration possible. Without version control, collaborating over source code would be a painful operation.

There are several perspectives on version control.

- For developers, it's a daily enabler for work and collaboration to happen. It's part of the daily job, one of the most-used tools.
- For management, the critical value of version control is in:
    - IP security.
    - Risk management.
    - Time-to-market speed through Continuous Delivery, where version control is a fundamental enabler.

Continue

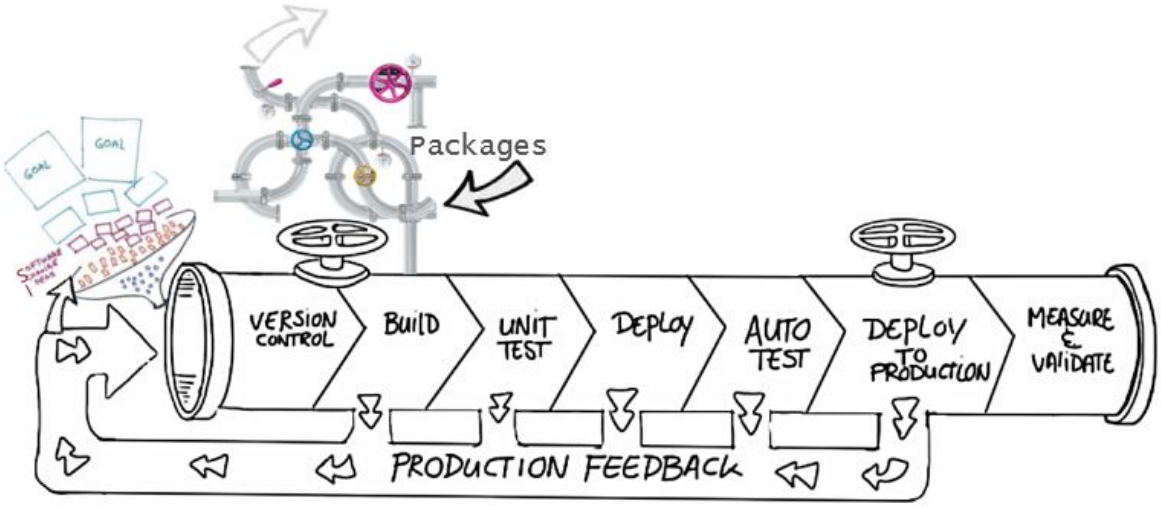
Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Explore benefits of source control

Completed

- 3 minutes

"Code does not exist unless it is committed into source control. Source control is the fundamental enabler of continuous delivery."



Whether writing code professionally or personally, you should always version your code using a source control management system. Some of the advantages of using source control are,

- **Create workflows** . Version control workflows prevent the chaos of everyone using their development process with different and incompatible tools. Version control systems provide process enforcement and permissions, so everyone stays on the same page.
- **Work with versions** . Every version has a description in the form of a comment. These descriptions help you follow changes in your code by version instead of by individual file changes. Code stored in versions can be viewed and restored from version control at any time as needed. It makes it easy to base new work on any version of code.
- **Collaboration** . Version control synchronizes versions and makes sure that your changes do not conflict with other changes from your team. Your team relies on version control to help resolve and prevent conflicts, even when people make changes simultaneously.
- **Maintains history of changes** . Version control keeps a record of changes as your team saves new versions of your code. This history can be reviewed to find out who, why, and when changes were made. The history gives you the confidence to experiment since you can roll back to a previous good version at any time. The history lets your base work from any code version, such as fixing a bug in an earlier release.
- **Automate tasks** . Version control automation features save your team time and generate consistent results. Automate testing, code analysis, and deployment when new versions are saved to version control.

Common software development values

- **Reusability** – why do the same thing twice? Reuse of code is a common practice and makes building on existing assets simpler.

- **Traceability** – Audits are not just for fun; in many industries, it is a legal matter. All activities must be traced, and managers can produce reports when needed. Traceability also makes debugging and identifying root cause easier. Additionally, it helps with feature reuse as developers can link requirements to implementation.
- **Manageability** – Can team leaders define and enforce workflows, review rules, create quality gates and enforce QA throughout the lifecycle?
- **Efficiency** – are we using the right resources for the job, minimizing time and effort? This one is self-explanatory.
- **Collaboration** – When teams work together, quality tends to improve. We catch one another's mistakes and can build on each other's strengths.
- **Learning** – Organizations benefit when they invest in employees learning and growing. It is important for onboarding new team members, the lifelong learning of seasoned members, and the opportunity for workers to contribute to the bottom line and the industry.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Explore best practices for source control

Completed

- 2 minutes

- **Make small changes** . In other words, commit early and commit often. Be careful not to commit any unfinished work that could break the build.
- **Do not commit personal files** . It could include application settings or SSH keys. Often personal files are committed accidentally but cause problems later when other team members work on the same code.
- **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 it will tell you why a change was made** . Consider committing messages as a mini form of documentation for the change.
- **Link code changes to work items** . It will concretely link what was created to why it was created—or modified by providing traceability across requirements and code changes.
- **No matter your background or preferences, be a team player and follow agreed conventions and workflows** . Consistency is essential and helps ensure quality, making it easier for team members to pick up where you left off, review your code, debug, and so on.

Using version control of some kind is necessary for any organization, and following the guidelines can help developers avoid needless time spent fixing errors and mistakes.

These practices also help organizations reap more significant benefits from having a good version control system.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Knowledge check

Completed

- 5 minutes

Choose the best response for each question. Then select **Check your answers** .

Check your knowledge

1.

Which of the following choices isn't a benefit of source control?

○

Manageability.

○

Efficiency.

○

Accountability.

2.

Which of the following choices isn't a source control best practice?

○

Make small changes.

○

Commit personal and secure files.

○

Link code changes to work items.

3.

Which of the following choices correctly describes one of the most valuable version control features?

Version control is a solution for managing and saving changes made to manually created assets. If you make changes to the source code, you can go back in time and easily roll back to previous-working versions.

Version control is a solution for automatically incrementing the version number for deployments.

Version control is a solution for managing and saving changes made to manually created assets. If you make changes to the source code, be careful, you can't go back in time and easily roll back to previous-working versions.

Check your answers

You must answer all questions before checking your work.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Summary

Completed

- 1 minute

This module explored the basics of source control and how to work with it daily to benefit team collaboration and code maintenance.

You learned how to describe the benefits and usage of:

- Understand source control.
- Apply best practices for source control.
- Describe the benefits of using source control.

Learn more

- Understand source control - Azure DevOps .
- Using source control in your codespace - GitHub Docs .

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Describe types of source control systems

Introduction

Completed

- 1 minute

This module describes different source control systems, such as Git and Team Foundation Version Control (TFVC), and helps with the initial steps for Git utilization.

Learning objectives

After completing this module, students and professionals can:

- Apply source control practices in your development process.
- Explain the differences between centralized and distributed version control.
- Understand Git and Team Foundation Version Control (TFVC).
- Develop using Git.

Prerequisites

- Understanding of what DevOps is and its concepts.
- Familiarity with version control principles is helpful but isn't necessary.
- Beneficial to have experience in an organization that delivers software.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Understand centralized source control

Completed

- 2 minutes

Strengths	Best used for
Easily scales for very large codebases	Large integrated codebases
Granular permission control	Audit & Access control down to file level
Permits monitoring of usage	Hard to merge file types
Allows exclusive file locking	

Centralized source control systems are based on the idea that there's a single "central" copy of your project somewhere (probably on a server). Programmers will check in (or commit) their changes to this central copy.

"Committing" a change means to record the difference in the central system. Other programmers can then see this change.

Also, it's possible to pull down the change. The version control tool will automatically update the contents of any files that were changed.

Most modern version control systems deal with "changesets," which are a group of changes (possibly too many files) that should be treated as a cohesive whole.

Programmers no longer must keep many copies of files on their hard drives manually. The version control tool can talk to the central copy and retrieve any version they need on the fly.

Some of the most common-centralized version control systems you may have heard of or used are Team Foundation Version Control (TFVC), CVS, Subversion (or SVN), and Perforce.

A typical centralized source control workflow

If working with a centralized source control system, your workflow for adding a new feature or fixing a bug in your project will usually look something like this:

- Get the latest changes other people have made from the central server.
- Make your changes, and make sure they work correctly.
- Check in your changes to the main server so that other programmers can see them.

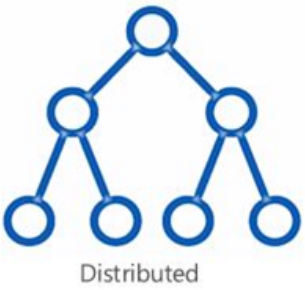
Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Understand distributed source control

Completed

- 3 minutes



Strengths	Best used for
Cross Platform support	Small & Modular codebases
An open source friendly code review model via pull requests	Evolving through open source
Complete offline support	Highly distributed teams
Portable history	Teams working across platforms
An enthusiastic growing user base	Green field codebases

Over time, so-called "distributed" source control or version control systems (DVCS for short) have become the most important.

The three most popular are Git, Mercurial, and Bazaar. These systems don't necessarily rely on a central server to store all the versions of a project's files. Instead, every developer "clones" a repository copy and has the project's complete history on their local storage. This copy (or "clone") has all the original metadata.

This method may sound wasteful but it isn't a problem in practice. Most programming projects consist primarily of plain text files (maybe a few images).

The disk space is so cheap that storing many copies of a file doesn't create a noticeable dent in a local storage free space. Modern systems also compress the files to use even less space; for example, objects (and deltas) are stored compressed, and text files used in programming compress well (around 60% of original size, or 40% reduction in size from compression).

Getting new changes from a repository is called "pulling." Moving your changes to a repository is called "pushing." You move changesets (changes to file groups as coherent wholes), not single-file diffs.

One common misconception about distributed version control systems is that there can't be a central project repository. It isn't true. Nothing stops you from saying, "this copy of the project is the authoritative one."

It means that instead of a central repository required by your tools, it's now optional.

Advantages over centralized source control

The act of cloning an entire repository gives distributed source control tools several advantages over centralized systems:

- Doing actions other than pushing and pulling changesets is fast because the tool only needs to access the local storage, not a remote server.

- Committing new changesets can be done locally without anyone else seeing them. Once you have a group of changesets ready, you can push all of them at once.
- Everything but pushing and pulling can be done without an internet connection. So, you can work on a plane, and you won't be forced to commit several bug fixes as one large changeset.
- Since each programmer has a full copy of the project repository, they can share changes with one, or two other people to get feedback before showing the changes to everyone.

Disadvantages compared to centralized source control

There are almost no disadvantages to using a distributed source control system over a centralized one.

Distributed systems don't prevent you from having a single "central" repository; they provide more options.

There are only two major inherent disadvantages to using a distributed system:

- If your project contains many large, binary files that can't be efficiently compressed, the space needed to store all versions of these files can accumulate quickly.
- If your project has a long history (50,000 changesets or more), downloading the entire history can take an impractical amount of time and disk space.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Explore Git and Team Foundation Version Control

Completed

- 2 minutes

Git (distributed)

Git is a distributed version control system. Each developer has a copy of the source repository on their development system. Developers can commit each set of changes on their dev machine.

Branches are lightweight. When you need to switch contexts, you can create a private local branch. You can quickly switch from one branch to another to pivot among different variations of your codebase. Later, you can merge, publish, or dispose of the branch.

Team Foundation Version Control (TFVC-centralized)

Team Foundation Version Control (TFVC) is a centralized version control system.

Typically, team members have only one version of each file on their dev machines. Historical data is maintained only on the server. Branches are path-based and created on the server.

TFVC has two workflow models:

- **Server workspaces** - Before making changes, team members publicly check out files. Most operations require developers to be connected to the server. This system helps lock workflows. Other software that works this way includes Visual Source Safe, Perforce, and CVS. You can scale up to huge codebases with millions of files per branch—also, large binary files with server workspaces.
- **Local workspaces** - Each team member copies the latest codebase version with them and works offline as needed. Developers check in their changes and resolve conflicts as necessary. Another system that works this way is Subversion.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Examine and choose Git

Completed

- 4 minutes

Switching from a centralized version control system to Git changes the way your development team creates software.

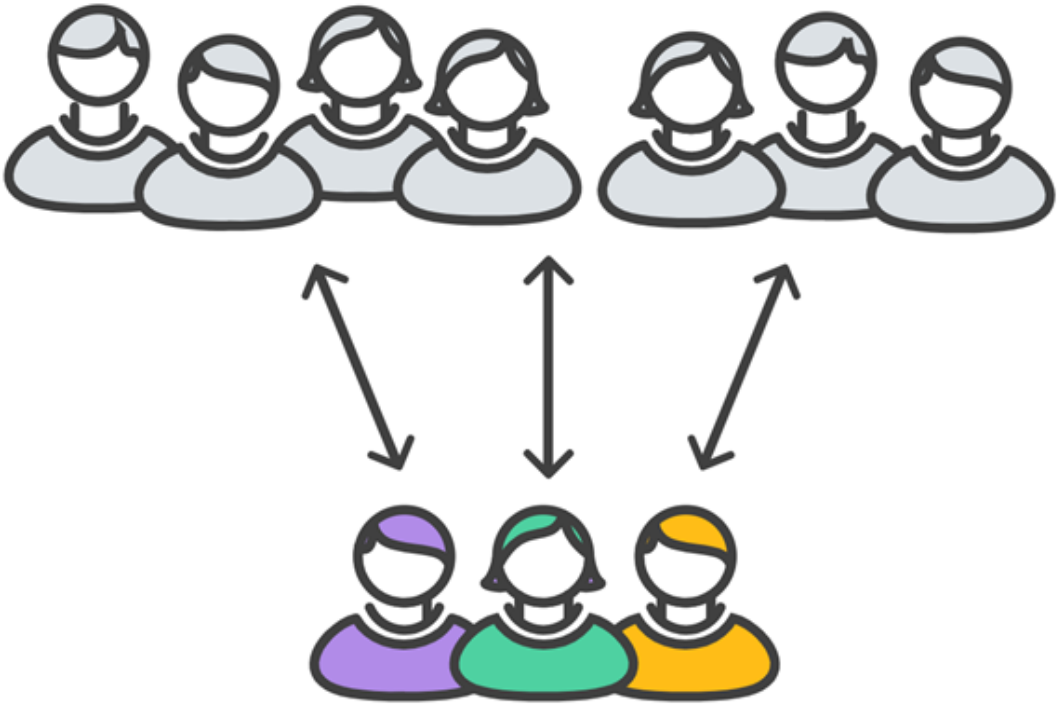
If you are a company that relies on its software for mission-critical applications, altering your development workflow impacts your entire business.

Developers would gain the following benefits by moving to Git.

Community

In many circles, Git has come to be the expected version control system for new projects.

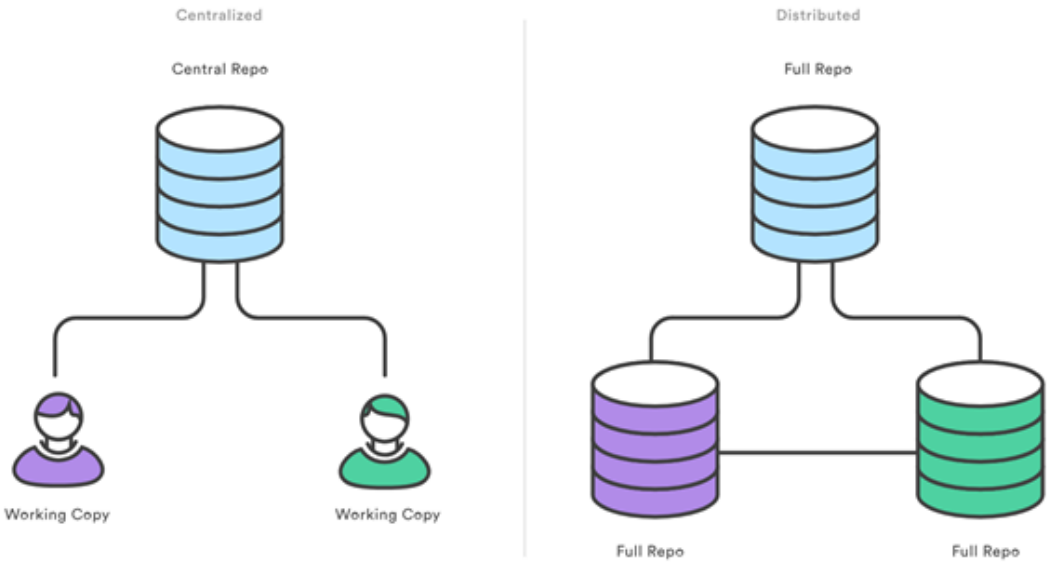
If your team is using Git, odds are you will not have to train new hires on your workflow because they will already be familiar with distributed development.



Also, Git is popular among open-source projects. It is easy to use 3rd-party libraries and encourage others to fork your open-source code.

Distributed development

In TFVC, each developer gets a working copy that points back to a single central repository. Git, however, is a distributed version control system. Instead of a working copy, each developer gets their local repository, complete with an entire history of commits.



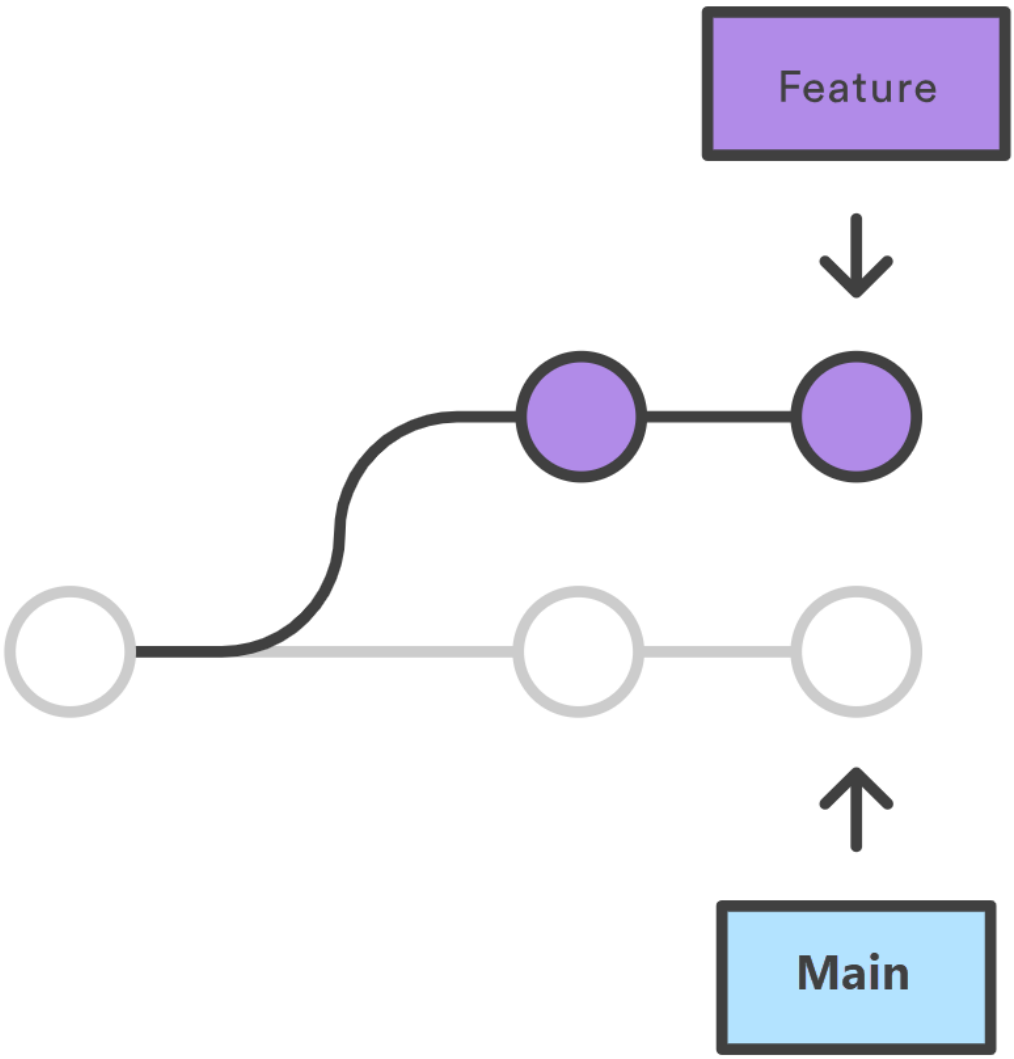
Having a complete local history makes Git fast since it means you do not need a network connection to create commits, inspect previous versions of a file, or do diffs between commits.

Distributed development also makes it easier to scale your engineering team. If someone breaks the production branch in SVN, other developers cannot check in their changes until it is fixed. With Git, this kind of blocking does not exist. Everybody can continue going about their business in their local repositories.

And, like feature branches, distributed development creates a more reliable environment. Even if developers obliterate their repository, they can clone from someone else and start afresh.

Trunk-based development

One of the most significant advantages of Git is its branching capabilities. Unlike centralized version control systems, Git branches are cheap and easy to merge.



Trunk-based development provides an isolated environment for every change to your codebase. When developers want to start working on something—no matter how large or small—they create a new branch. It ensures that the main branch always contains production-quality code.

Using trunk-based development is more reliable than directly-editing production code, but it also provides organizational benefits.

They let you represent development work at the same granularity as your agile backlog.

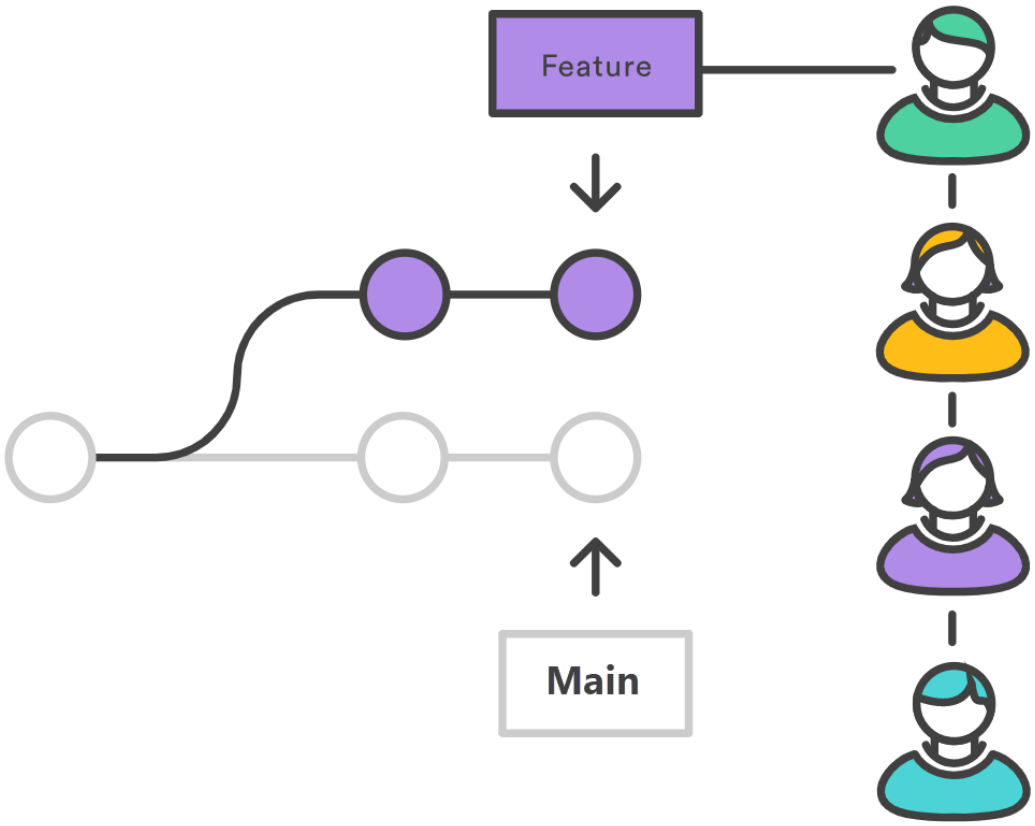
For example, you might implement a policy where each work item is addressed in its feature branch.

Pull requests

Many source code management tools such as Azure Repos enhance core Git functionality with pull requests.

A pull request is a way to ask another developer to merge one of your branches into their repository.

It makes it easier for project leads to keep track of changes and lets developers start discussions around their work before integrating it with the rest of the codebase.



Since they are essentially a comment thread attached to a feature branch, pull requests are incredibly versatile.

When a developer gets stuck with a complex problem, they can open a pull request to ask for help from the rest of the team.

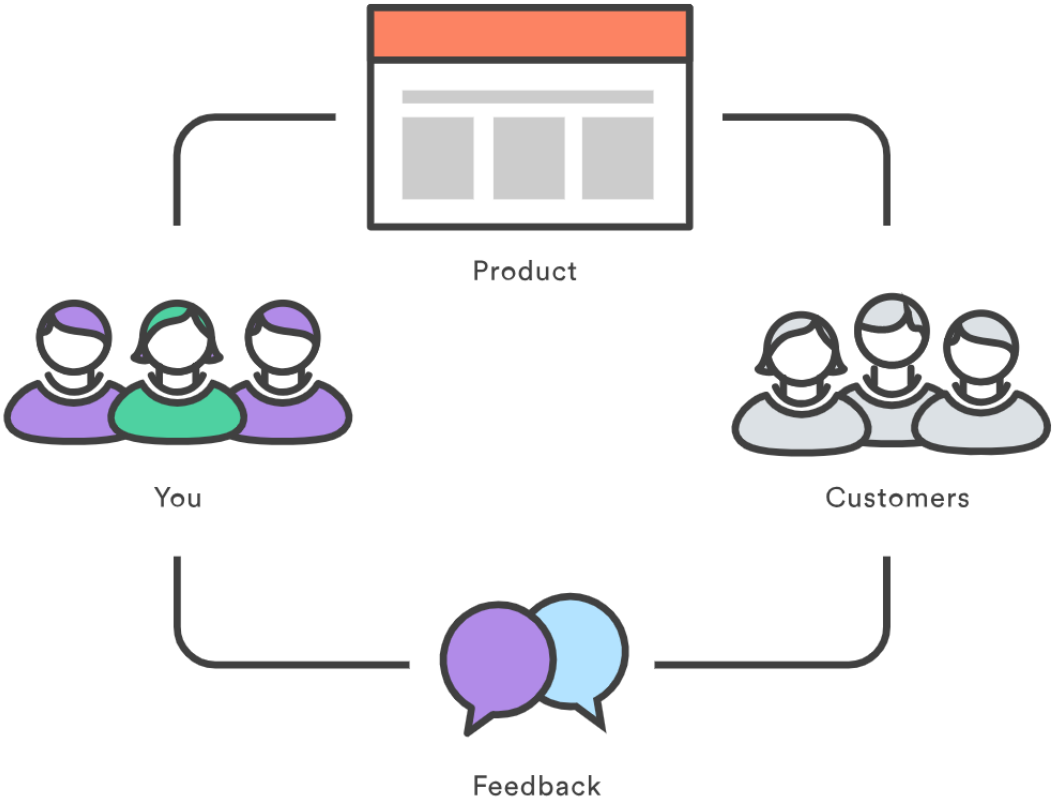
Instead, junior developers can be confident that they are not destroying the entire project by treating pull requests as a formal code review.

Faster release cycle

A faster release cycle is the ultimate result of feature branches, distributed development, pull requests, and a stable community.

These capabilities promote an agile workflow where developers are encouraged to share more minor changes more frequently.

In turn, changes can get pushed down the deployment pipeline faster than the standard of the monolithic releases with centralized version control systems.



As you might expect, Git works well with continuous integration and continuous delivery environments.

Git hooks allow you to run scripts when certain events occur inside a repository, which lets you automate deployment to your heart's content.

You can even build or deploy code from specific branches to different servers.

For example, you might want to configure Git to deploy the most recent commit from the develop branch to a test server whenever anyone merges a pull request into it.

Combining this kind of build automation with peer review means you have the highest possible confidence in your code as it moves from development to staging to production.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Understand objections to using Git

Completed

- 2 minutes

There are three common objections I often hear to migrating to Git:

- I can overwrite history.
- I have large files.
- There is a steep learning curve.

Overwriting history

Git technically does allow you to overwrite history - but like any helpful feature, if misused can cause conflicts.

If your teams are careful, they should never have to overwrite history.

If you are synchronizing to Azure Repos, you can also add a security rule that prevents developers from overwriting history by using the explicit "Force Push" permissions.

Every source control system works best when developers understand how it works and which conventions work.

While you cannot overwrite history with Team Foundation Version Control (TFVC), you can still overwrite code and do other painful things.

Large files

Git works best with repos that are small and do not contain large files (or binaries).

Every time you (or your build machines) clone the repo, they get the entire repo with its history from the first commit.

It is great for most situations but can be frustrating if you have large files.

Binary files are even worse because Git cannot optimize how they are stored.

That is why Git LFS was created.

It lets you separate large files of your repos and still has all the benefits of versioning and comparing.

Also, if you are used to storing compiled binaries in your source repos, stop!

Use Azure Artifacts or some other package management tool to store binaries for which you have source code.

However, teams with large files (like 3D models or other assets) can use Git LFS to keep the code repo slim and trimmed.

Learning curve

There is a learning curve. If you have never used source control before, you are probably better off when learning Git. I have found that users of centralized source control (TFVC or SubVersion) battle initially to make the mental shift, especially around branches and synchronizing.

Once developers understand how Git branches work and get over the fact that they must commit and then push, they have all the basics they need to succeed in Git.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Describe working with Git locally

Completed

- 7 minutes

Git and Continuous Delivery is one of those delicious chocolate and peanut butter combinations. We occasionally find two great tastes that taste great together in the software world!

Continuous Delivery of software demands a significant level of automation. It is hard to deliver continuously if you do not have a quality codebase.

Git provides you with the building blocks to take charge of quality in your codebase. It allows you to automate most of the checks in your codebase.

Also, it works before committing the code into your repository.

To fully appreciate the effectiveness of Git, you must first understand how to carry out basic operations on Git. For example, clone, commit, push, and pull.

The natural question is, how do we get started with Git?

One option is to go native with the command line or look for a code editor that supports Git natively.

Visual Studio Code is a cross-platform, open-source code editor that provides powerful developer tooling for hundreds of languages.

To work in open-source, you need to embrace open-source tools.

This recipe will start by:

- Setting up the development environment with Visual Studio Code.
- Creating a new Git repository.
- Committing code changes locally.
- Pushing changes to a remote repository on Azure DevOps.

Getting ready

This tutorial will teach us how to initialize a Git repository locally.

Then we will use the ASP.NET Core MVC project template to create a new project and version it in the local Git repository.

We will then use Visual Studio Code to interact with the Git repository to do basic commit, pull, and push operations.

You will need to set up your working environment with the following:

- .NET Core 3.1 SDK or later: Download .NET .
- Visual Studio Code: Download Visual Studio Code .
- C# Visual Studio Code extension: C# programming with Visual Studio Code .
- Git: Git - Downloads
- Git for Windows (if you are using Windows): Git for Windows

The Visual Studio Marketplace features several extensions for Visual Studio Code that you can install to enhance your experience of using Git:

- Git Lens : This extension brings visualization for code history by using Git blame annotations and code lens. The extension enables you to seamlessly navigate and explore the history of a file or branch. Also, the extension allows you to gain valuable insights via powerful comparison commands and much more.
- Git History : Brings visualization and interaction capabilities to view the Git log, file history and compare branches or commits.

How to do it

1. Open the Command Prompt and create a new-working folder:

```
mkdir myWebApp
cd myWebApp
```

2. In myWebApp, initialize a new Git repository:

```
git init
```

3. Configure global settings for the name and email address to be used when committing in this Git repository:

```
git config --global user.name "John Doe"
git config --global user.email "john.doe@contoso.com"
```

If you are working behind an enterprise proxy, you can make your Git repository proxy-aware by adding the proxy details in the Git global configuration file.

Different variations of this command will allow you to set up an HTTP/HTTPS proxy (with username/password) and optionally bypass SSL verification.

Run the below command to configure a proxy in your global git config.

```
git config --global http.proxy
http://proxyUsername:proxyPassword@proxy.server.com:port
```

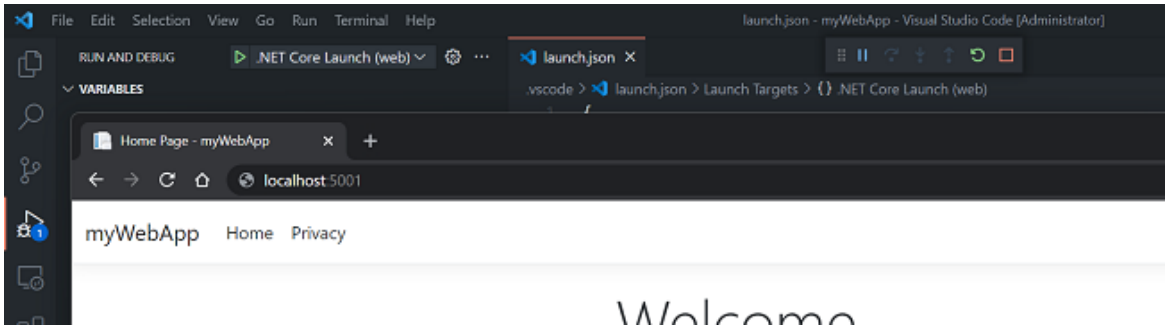
4. Create a new ASP.NET core application. The new command offers a collection of switches that can be used for language, authentication, and framework selection. More details can be found on [Microsoft docs](#).

```
dotnet new mvc
```

Launch Visual Studio Code in the context of the current-working folder:

```
code .
```

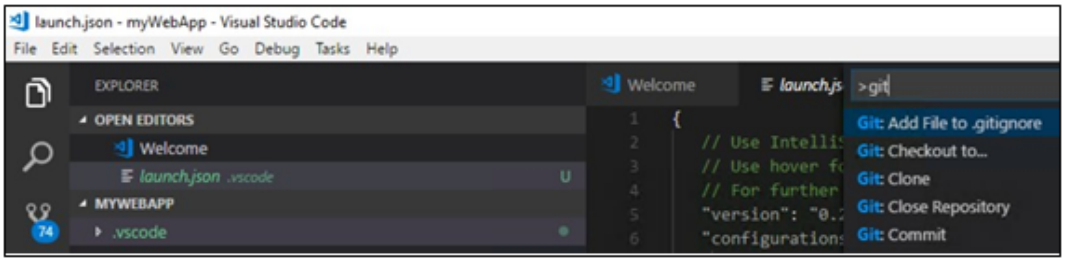
5. When the project opens in Visual Studio Code, select **Yes** for the **Required assets to build and debug are missing from 'myWebApp.' Add them?** Warning message. Select **Restore** for the **There are unresolved dependencies** info message. Hit **F5** to debug the application, then myWebApp will load in the browser, as shown in the following screenshot:



If you prefer to use the command line, you can run the following commands in the context of the git repository to run the web application.

```
dotnet build
dotnet run
```

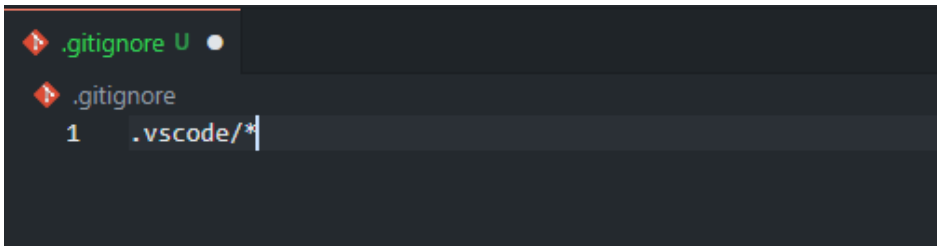
You will notice the ".vscode" folder is added to your working folder. To avoid committing this folder to your Git repository, you can include it in the .gitignore file. Select a file from the ".vscode" folder, hit F1 to launch the command window in Visual Studio Code, type gitIgnore, and accept the option to include the selected file in the new .gitIgnore file.



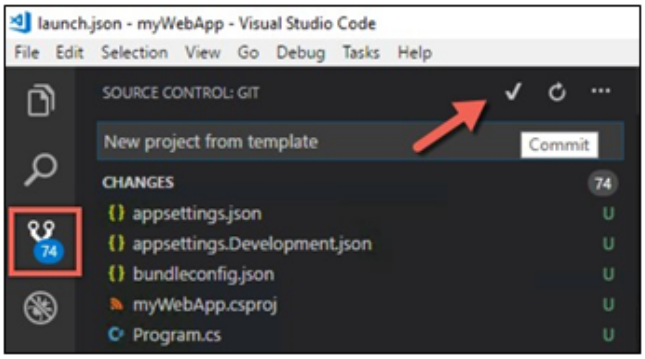
Note

To ignore an entire directory, you need to include the name of the directory with the slash / at the end.

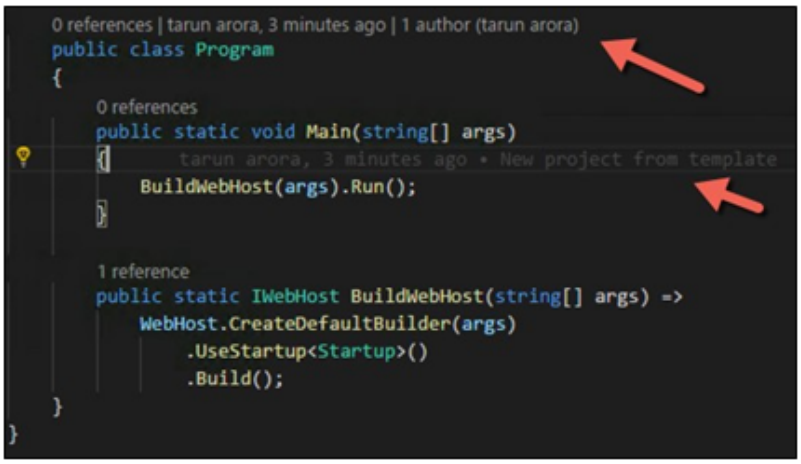
Open your .gitignore, remove the file name from the path, and leave the folder with a slash, for example, *.vscode/** .



6. To stage and commit the newly created myWebApp project to your Git repository from Visual Studio Code, navigate the Git icon from the left panel. Add a commit comment and commit the changes by clicking the checkmark icon. It will stage and commit the changes in one operation:



Open Program.cs, you will notice Git lens decorates the classes and functions with the commit history and brings this information in line to every line of code:



7. Now launch cmd in the context of the git repository and run `git branch --list` . It will show you that only the `main` branch currently exists in this repository. Now run the following command to create a new branch called `feature-devops-home-page` .

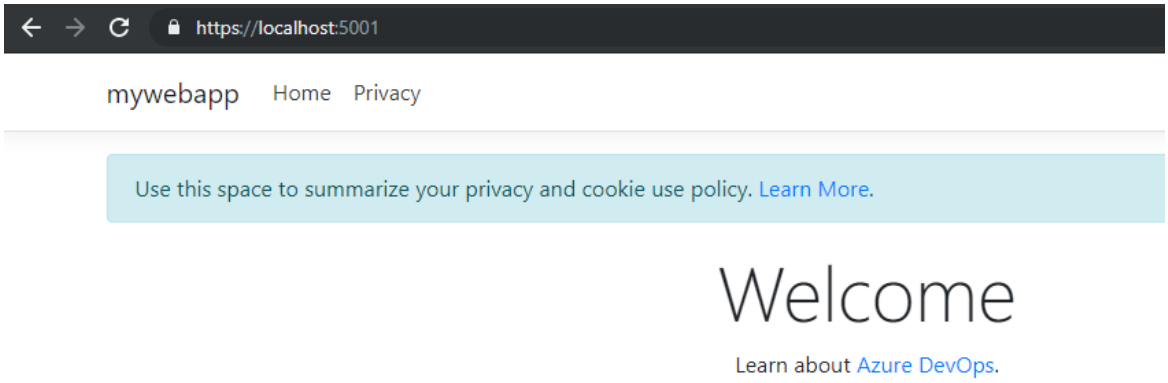
```
git branch feature-devops-home-page
git checkout feature-devops-home-page
git branch --list
```

   You have created a new branch with these commands and checked it out. The `--list` keyword shows you a list of all branches in your repository. The green color represents the branch that is currently checked out.

8. Now navigate to the file `~\Views\Home\Index.cshtml` and replace the contents with the text below.

```
@{
    ViewData["Title"] = "Home Page";
}

<div class="text-center">
    <h1 class="display-4">Welcome</h1>
    <p>Learn about <a href="https://azure.microsoft.com/services/devops/">Azure
DevOps</a>.</p>
</div>
```

9. Refresh the web app in the browser to see the changes.

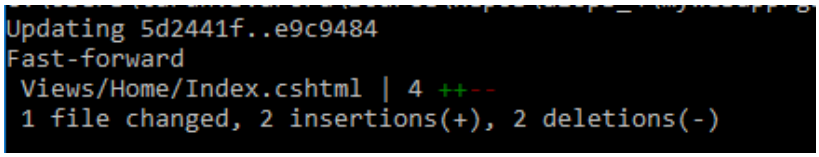


10. In the context of the git repository, execute the following commands. These commands will stage the changes in the branch and then commit them.

```
git status

git add .

git commit -m "updated welcome page."

git status
```

11. To merge the changes from the feature-devops-home-page into the main, run the following commands in the context of the git repository.

```
git checkout main

git merge feature-devops-home-page
```



12. Run the below command to delete the feature branch.

```
git branch --delete feature-devops-home-page
```

How it works

The easiest way to understand the outcome of the steps done earlier is to check the history of the operation. Let us have a look at how to do it.

1. In Git, committing changes to a repository is a two-step process. Running: `add .` The changes are staged but not committed. Finally, running the commit promotes the

staged changes in the repository.

2. To see the history of changes in the main branch, run the command `git log -v`

```
commit e9c948427c1aa99e8aede67f6a2be206d148beaf
Author: Tarun Arora <tarun.arora@contoso.com>
Date:   Thu Jul 25 12:45:43 2019 +0100

    updated welcome page

commit 5d2441f0be4f1e4ca1f8f83b56dee31251367adc
Author: Tarun Arora <tarun.arora@contoso.com>
Date:   Thu Jul 25 12:07:55 2019 +0100

    project init
```

3. To investigate the actual changes in the commit, you can run the command `git log -p`

```
commit e9c948427c1aa99e8aede67f6a2be206d148beaf
Author: Tarun Arora <tarun.arora@contoso.com>
Date:   Thu Jul 25 12:45:43 2019 +0100

    updated welcome page

diff --git a/Views/Home/Index.cshtml b/Views/Home/Index.cshtml
index d2d19bd..6d8ad94 100644
--- a/Views/Home/Index.cshtml
+++ b/Views/Home/Index.cshtml
@@ -4,5 +4,5 @@

 <div class="text-center">
     <h1 class="display-4">Welcome</h1>
-    <p>Learn about <a href="https://docs.microsoft.com/aspnet/core">building Web apps with ASP.NET Core</a>.</p>
-</div>
+    <p>Learn about <a href="https://azure.microsoft.com/en-gb/services/devops/">Azure DevOps</a>.</p>
+</div>
\ No newline at end of file
```

There is more

Git makes it easy to back out changes. Following our example, if you want to take out the changes made to the welcome page.

You can do It hard resetting the main branch to a previous version of the commit using the following command.

```
git reset --hard 5d2441f0be4f1e4ca1f8f83b56dee31251367adc
```

Running the above command would reset the branch to the project init change.

If you run `git log -v`, you will see that the changes done to the welcome page are removed from the repository.

[Continue]

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Knowledge check

Completed

- 4 minutes

Choose the best response for each question. Then select **Check your answers** .

Check your knowledge

1.

Which of the following choices correctly describes what is source control?

○

Source control is the practice of controlling what is deployed to test and production environments.

○

Source control is the practice of controlling security through code files.

○

Source control is the practice of tracking and managing changes to code.

2.

Which of the following choices isn't a benefit of using distributed version control?

○

Complete offline support

○

Allows exclusive file locking.

○

Portable history.

3.

Which of the following choices isn't a benefit of using centralized version control?

○

Easily scales for large codebases.

An open-source friendly code review model via pull requests.

Allows exclusive file locking.

Check your answers

You must answer all questions before checking your work.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Summary

Completed

- 1 minute

This module described different source control systems such Git and Team Foundation Version Control (TFVC) and helped with the initial steps for Git utilization.

You learned how to describe the benefits and usage of:

- Apply source control practices in your development process.
- Explain the differences between centralized and distributed version control.
- Understand Git and Team Foundation Version Control (TFVC).
- Develop using Git.

Learn more

- What is Team Foundation Version Control - Azure Repos | Microsoft Docs .
- Migrate from TFVC to Git - Azure DevOps | Microsoft Docs .
- Git and TFVC version control - Azure Repos | Microsoft Docs .
- Get started with Git and Visual Studio - Azure Repos | Microsoft Docs .

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Work with Azure Repos and GitHub

Introduction

Completed

- 1 minute

This module introduces Azure Repos and GitHub and explores ways to migrate from TFVC to Git and work with GitHub Codespaces for development.

Learning objectives

After completing this module, students and professionals can:

- Describe Azure Repos and GitHub.
- Migrate from TFVC to Git.
- Work with GitHub Codespaces.

Prerequisites

- Understanding of what DevOps is and its concepts.
- Familiarity with Git and version control principles is helpful but is not necessary.
- Beneficial to have experience in an organization that delivers software.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Introduction to Azure Repos

Completed

- 2 minutes

Azure Repos is a set of version control tools that you can use to manage your code.

Using version control is a good idea whether your software project is large or small.

Azure Repos provides two types of version control:

- Git: distributed version control
- Team Foundation Version Control (TFVC): centralized version control

What do I get with Azure Repos?

- Use free private Git repositories, pull requests, and code search: Get unlimited private Git repository hosting and support for TFVC that scales from a hobby project to the world's largest repository.
- Support for any Git client: Securely connect with and push code into your Git repository from any IDE, editor, or Git client.
- Web hooks and API integration: Add validations and extensions from the marketplace or build your own-using web hooks and REST APIs.
- Semantic code search: Quickly find what you are looking for with a code-aware search that understands classes and variables.
- Collab to build better code: Do more effective Git code reviews with threaded discussion and continuous integration for each change. Use forks to promote collaboration with inner source workflows.
- Automation with built-in CI/CD: Set up continuous integration/continuous delivery (CI/CD) to automatically trigger builds, tests, and deployments. Including every completed pull request using Azure Pipelines or your tools.
- Protection of your code quality with branch policies: Keep code quality high by requiring code reviewer sign-out, successful builds, and passing tests before merging pull requests. Customize your branch policies to maintain your team's high standards.
- Usage of your favorite tools: Use Git and TFVC repositories on Azure Repos with your favorite editor and IDE.

For further reference on using git in Azure Repos, refer to Microsoft Docs.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Introduction to GitHub

Completed

- 3 minutes

GitHub is the largest open-source community in the world. Microsoft owns GitHub. GitHub is a development platform inspired by the way you work.

You can host and review code, manage projects, and build software alongside 40 million developers from open source to business.

GitHub is a Git repository hosting service that adds many of its features.

While Git is a command-line tool, GitHub provides a Web-based graphical interface.

It also provides access control and several collaboration features, such as wikis and essential task management tools for every project.

So what are the main benefits of using GitHub? Nearly every open-source project uses GitHub to manage its project.

Using GitHub is free if your project is open source and includes a wiki and issue tracker, making it easy to have more in-depth documentation and get feedback about your project.

What are some of the features offered by GitHub?

- Automate from code to cloud: Cycle your production code faster and simplify your workflow with GitHub Packages and built-in CI/CD using GitHub Actions.
    - Automate your workflows: Build, test, deploy, and run CI/CD how you want in the same place you manage code. Trigger Actions from any GitHub event to any available API. Build your Actions in the language of your choice, or choose from thousands of workflows and Actions created by the community.
    - Packages at home with their code: Use Actions to automatically publish new package versions to GitHub Packages. Install packages and images hosted on GitHub Packages or your preferred packages registry in your CI/CD workflows. It is always free for open source, and data transfer within Actions is unlimited for everyone.
- Securing software together: GitHub plays a role in securing the world's code—developers, maintainers, researchers, and security teams. On GitHub, development teams everywhere can work together to secure the world's software supply chain, from fork to finish.
    - Get alerts about vulnerabilities in your code: GitHub continuously scans security advisories for popular languages. Also, it sends security alerts to maintainers of affected repositories with details so they can remediate risks.
    - Automatically update vulnerabilities: GitHub monitors your project dependencies and automatically opens pull requests to update dependencies to the minimum version that resolves known vulnerabilities.
    - Stay on top of CVEs: Stay updated with the latest Common Vulnerabilities and Exposures (CVEs), and learn how they affect you with the GitHub Advisory Database.
    - Find vulnerabilities that other tools miss: CodeQL is the industry's leading semantic code analysis engine. GitHub's revolutionary approach treats code as data to identify security vulnerabilities faster.
    - Eliminate variants: Never make the same mistake twice. Proactive vulnerability scanning prevents vulnerabilities from ever reaching production.
    - Keep your tokens safe: Accidentally commit a token to a public repository? GitHub got you. With support from 20 service providers, GitHub takes steps to keep you safe.
- Seamless code review: Code review is the surest path to better code and is fundamental to how GitHub works. Built-in review tools make code review an essential part of your team's process.

    - Propose changes: Better code starts with a Pull Request, a living conversation about changes where you can talk through ideas, assign tasks, discuss details, and conduct reviews.
    - Request reviews: If you are on the other side of a review, you can request reviews from your peers to get the detailed feedback you need.
    - See the difference: Reviews happen faster when you know exactly what changes. Diffs compare versions of your source code, highlighting the new, edited, or deleted parts.
    - Comment in context: Discussions happen in comment threads within your code—bundle comments into one review or reply to someone else who is in line to start a conversation.
    - Give clear feedback: Your teammates should not have to think too hard about what a thumbs-up emoji means. Specify whether your comments are required changes or just a few suggestions.
    - Protect branches: Only merge the highest-quality code. You can configure repositories to require status checks, reducing human error and administrative overhead.

- All your code and documentation in one place: Hundreds of millions of private, public, and open-source repositories are hosted on GitHub. Every repository has tools to help you host, version, and release code and documentation.

    - Code where you collaborate: Repositories keep code in one place and help your teams collaborate with the tools they love, even if you work with large files using Git LFS. You can create or import as many projects as possible with unlimited private repositories for individuals and groups.
    - Documentation alongside your code: Host your documentation directly from your repositories with GitHub Pages. Use Jekyll as a static site generator and publish your Pages from the /docs folder on your main branch.

- Manage your ideas: Coordinate early, stay aligned, and get more done with GitHub's project management tools.

    - See your project's large picture: See everything happening in your project and choose where to focus your team's efforts with Projects and task boards that live right where they belong: close to your code.
    - Track and assign tasks: Issues help you identify, assign, and keep track of tasks within your team. You can open an Issue to track a bug, discuss an idea with an @mention , or start distributing work.

- The human side of software: Building software is more about managing teams and communities than coding. Whether on a group of two or 2000, GitHub has the support your people need.

    - Manage and grow teams: Help people organize with GitHub teams, level up to access administrative roles, and fine-tune your permissions with nested teams.

- Keep conversations: Moderation tools, like issue and pull request locking, help your team stay focused on code. And if you maintain an open-source project, user blocking reduces noise and ensures productive conversations.
- Set community guidelines: Set roles and expectations without starting from scratch. Customize standard codes of conduct to create the perfect one for your project. Then choose a pre-written license right from your repository.

GitHub offers excellent learning resources for its platform. You can find everything from git introduction training to deep dive on publishing static pages to GitHub and how to do DevOps on GitHub right here .

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Migrate from TFVC to Git

Completed

- 2 minutes

Migrating the tip

Most teams wish they could reorganize their source control structure.

Typically, the structure the team is using today was set up by a well-meaning developer a decade ago, but it is not optimal.

Migrating to Git could be an excellent opportunity to restructure your repo.

In this case, it probably does not make sense to migrate history anyway since you are going to restructure the code (or break the code into multiple repos).

The process is simple:

- Create an empty Git repo (or multiple empty repos).
- Get-latest from TFS.
- Copy/reorganize the code into the empty Git repos.
- Commit and push, and you are there!

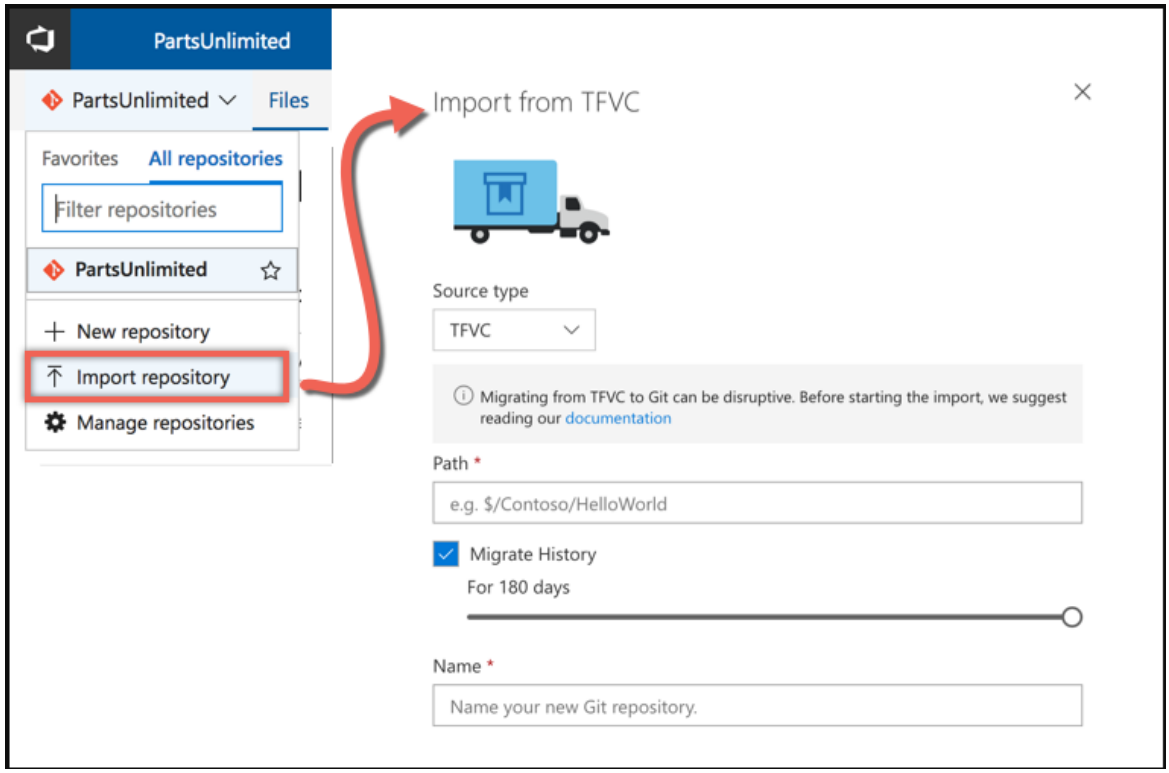
If you have shared code, you need to create builds of the shared code to publish to a package feed. And then consume those packages in downstream applications, but the Git part is straightforward.

Single branch import

If you are on TFVC and in Azure DevOps, you have the option of a simple single-branch import. Click on the Import repository from the Azure Repos top-level drop-down menu to open the dialog. Then enter the path to the branch you are migrating to (yes, you can only choose one branch). Select if you want history or not (up to 180 days). Add in a name for the repo, and the import will be triggered.

Import repository

Import repository also allows you to import a git repository. It is beneficial to move your git repositories from GitHub or any other public or private hosting spaces into Azure Repos.



There are some limitations here (that apply only when migrating source type TFVC): a single branch and only 180 days of history.

However, if you only care about one branch and are already in Azure DevOps, it is an effortless but effective way to migrate.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Use GIT-TFS

Completed

- 3 minutes

What if you need to migrate more than a single branch and keep branch relationships? Or are you going to drag all the history with you?

In that case, you're going to have to use **GIT-TFS** . It's an open-source project built to synchronize Git and TFVC repositories.

But you can do a once-off migration using the Git TFS clone.

GIT-TFS has the advantage that it can migrate multiple branches and preserve the relationships to merge branches in Git after you migrate.

Be warned that doing this conversion can take a while - especially for large or long-history repositories.

You can quickly dry-run the migration locally, iron out any issues, and then do it for real. There are lots of flexibilities with this tool.

If you are on Subversion, you can use **GIT-SVN** to import your Subversion repo similarly to **GIT-TFS** .

Migrating from TFVC to Git using GIT-TFS

If Chocolatey is already installed on your computer, run `choco install gittfs`

Add the GIT-TFS folder path to your PATH. You could also set it temporary (the time of your current terminal session) using: `set PATH=%PATH%;%cd%\GitTfs\bin\Debug`

You need .NET 4.5.2 and maybe the 2012 or 2013 version of Team Explorer (or Visual Studio). It depends on the version of Azure DevOps you want to target.

Clone the whole repository (wait for a while.) :

```
git tfs clone http://tfs:8080/tfs/DefaultCollection $/some_project
```

Advanced use cases of cloning the TFVC repository into Git are documented here .

```
cd some_project
git log
```

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Develop online with GitHub Codespaces

Completed

- 2 minutes

GitHub Codespaces addresses several issues from which developers regularly suffer.

First, many developers are working with old hardware and software systems, which are not refreshed.

Second, developers are often tied to individual development systems. Moving from location to location or system to system is inconvenient or slow to configure.

A problem for the developers' organizations is the proliferation of intellectual property across all these machines.

What is GitHub Codespaces?

Codespaces is a cloud-based development environment that GitHub hosts. It is essentially an online implementation of Visual Studio Code.

Codespaces allows developers to work entirely in the cloud.

Codespaces even will enable developers to contribute from tablets and Chromebooks.

Because it is based on Visual Studio Code, the development environment is still rich with:

- Syntax highlighting.
- Autocomplete.
- Integrated debugging.
- Direct Git integration.

Developers can create a codespace (or multiple codespaces) for a repository. Each codespace is associated with a specific branch of a repository.

Using GitHub Codespaces

You can do all your work in codespaces within a browser.

For an even more responsive experience, you can connect to a codespace from a local copy of Visual Studio Code.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Knowledge check

Completed

- 4 minutes

Choose the best response for each question. Then select **Check your answers** .

Check your knowledge

1.

Which of the following choices isn't a supported Azure Repos version control system?

Git.

Team Foundation Version Control (TFVC).

Source Safe.

2.

Which of the following choices is the exact number of days of history you can import Team Foundation Version Control (TFVC) to Git using the "Import repository" feature in Azure DevOps?

90 days.

180 days.

365 days.

3.

Which of the following choices describe GitHub Codespaces?

○

It's a platform for hosting and managing packages, including containers and other dependencies.

○

It's an AI pair programmer that helps you write code faster and with less work.

○

It's an online implementation of Visual Studio Code.

Check your answers

You must answer all questions before checking your work.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Summary

Completed

- 1 minute

This module introduced Azure Repos and GitHub and explored ways to migrate from TFVC to Git and work with GitHub Codespaces for development.

You learned how to describe the benefits and usage of:

- Describe Azure Repos and GitHub.
- Migrate from TFVC to Git.
- Work with GitHub Codespaces.

Learn more

- Integration of Azure Repos and Git Repositories .
- Integration of Azure Boards and GitHub .
- Import repositories from TFVC to Git - Azure Repos | Microsoft Docs .
- GitHub Codespaces .

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .