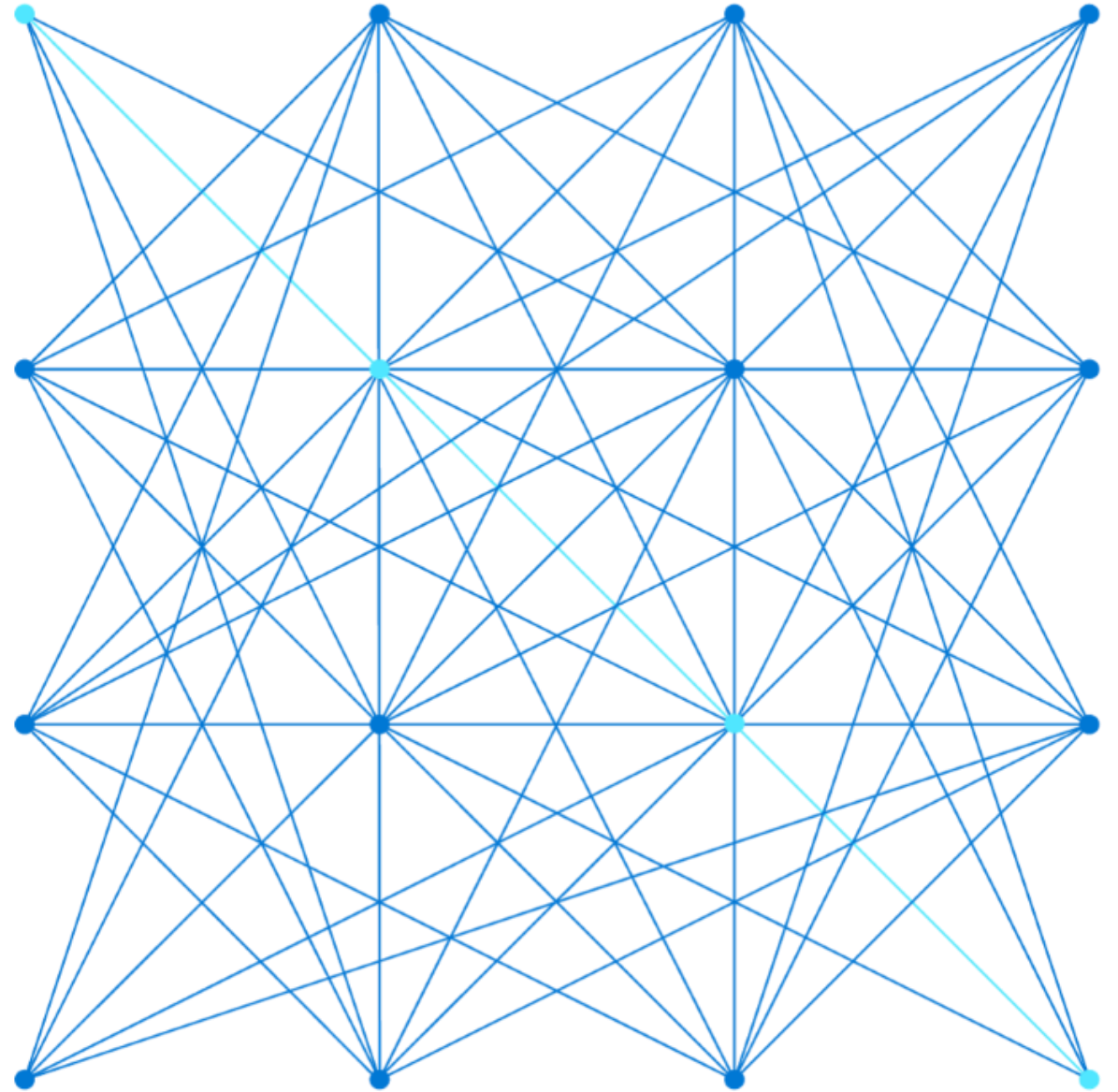


AZ-400.00

Module 1: Planning for DevOps



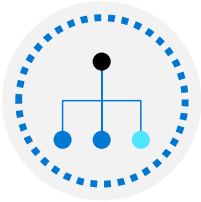
Lesson 01: Module overview



Module overview



Lesson 1: Module Overview



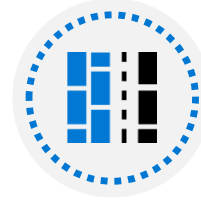
Lesson 2: Transformation Planning



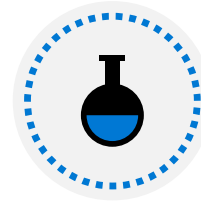
Lesson 3: Project Selection



Lesson 4: Team Structures



Lesson 5: Migrating to Azure DevOps



Lesson 6: Lab



Lesson 7: Module Review and Takeaways

Learning objectives

After completing this module, students will be able to:



Plan for the transformation with shared goals and timelines



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



Create a team and agile organizational structure



Design a tool integration strategy



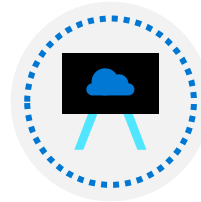
Design a license management strategy (e.g. Azure DevOps users)



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

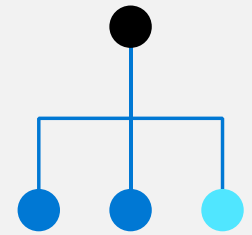


Design an authentication and access strategy



Design a strategy for integrating on-premises and cloud resources

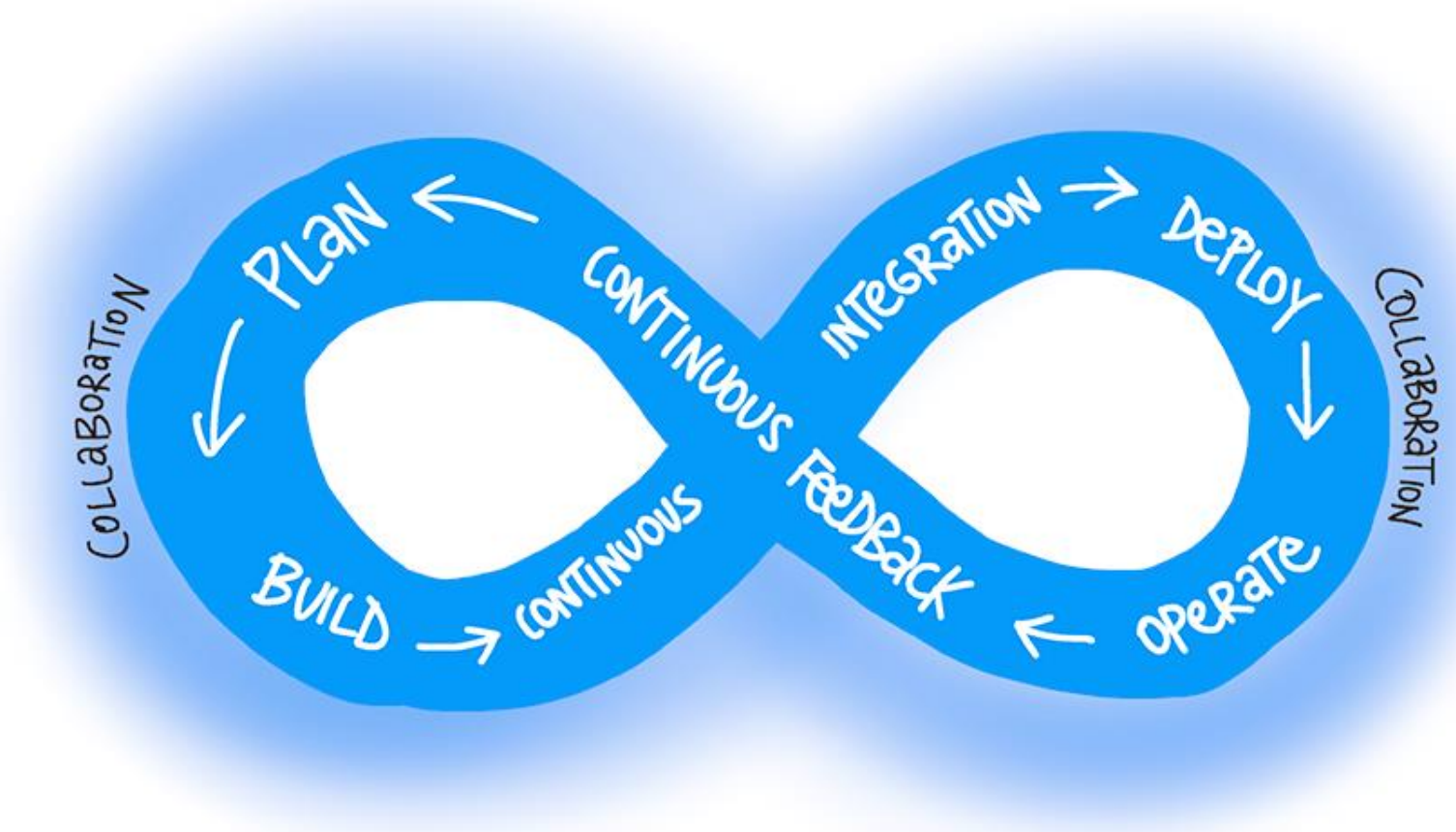
Lesson 02: Transformation planning



What is DevOps?

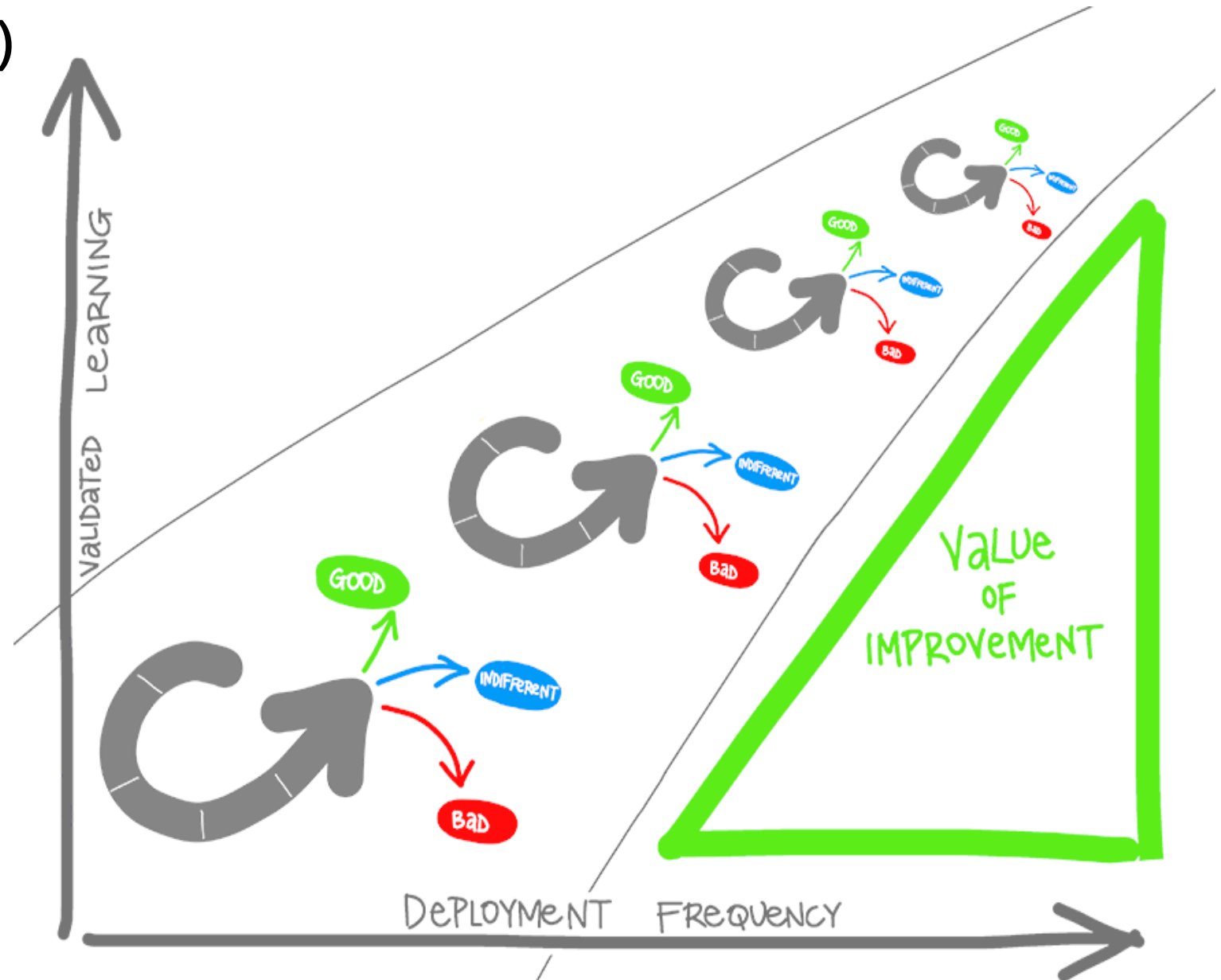
"DevOps is the union of people, process, and products to enable continuous delivery of value to end users."

– Donovan Brown, [What is DevOps?](#)



What is DevOps? (continued)

- Understand your cycle time
 - Observe, Orient, Decide, Act (OODA) loop
- Become data-informed
- Strive for validated learning
- Shorten your cycle time
- Optimize validated learning



The DevOps journey

- Continuous Integration
- Continuous Delivery
- Version Control
- Agile/lean
- Monitoring and logging
- Cloud
- Infrastructure as Code (IaC)
- Microservices
- Containers
- DevOps may hurt at first

<https://docs.microsoft.com/en-us/azure/devops/learn/what-is-devops>

Separating transformation teams



There are several challenges when creating teams:

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



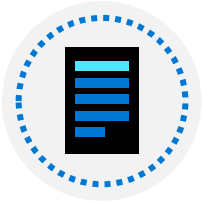
To overcome the challenges, create a team that is:

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



A transformation project can conflict with ongoing business needs

Defining shared goals



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

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



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

Setting timelines for goals



Measurable goals should have timelines that challenging yet achievable

(#mw) **S**pecific **M**easurable **A**ttainable **R**elevant **T**ime-bound



Timelines should be a constant series of short-term goals – each clear and measurable



Shorter timelines have advantages:

- Easier to change plans or priorities when necessary
- Reduced delay between doing the work and getting feedback
- Easier to keep organizational support when positive outcomes are apparent

Lesson 03: Project selection

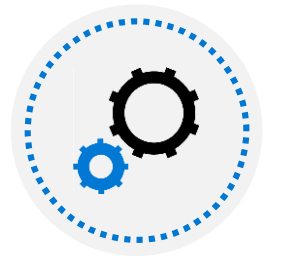


Greenfield and brownfield projects defined

Greenfield software projects develop in a totally new environment.



Brownfield software projects develop in the immediate presence of existing software applications/systems.

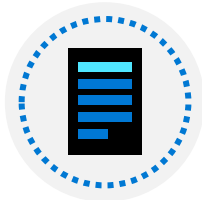


Choosing greenfield and brownfield projects



Greenfield projects:

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



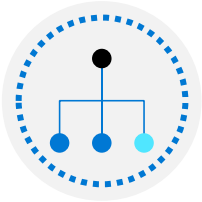
Brownfield projects:

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



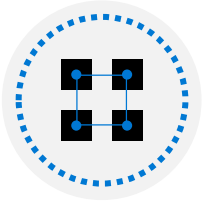
There is a common misconception that DevOps suits Greenfield projects better than Brownfield projects, but this is not the case

Choosing systems of record versus systems of engagement (as a DevOps Pilot)



Systems of record:

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



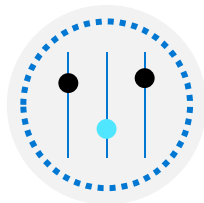
Systems of engagement:

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



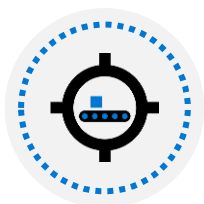
Both types of systems are important

Selecting groups to minimize initial resistance



Different types of staff members:

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

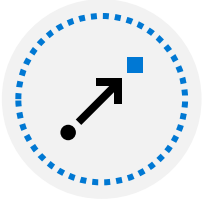


Ideal Target Improvements:

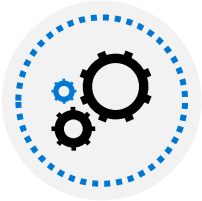
- Can be used to gain early wins
- Are small enough to be achievable in a reasonable time-frame
- Have benefits that are significant enough to be obvious to the organization

(#mw) we're looking for quickly realised successful outcomes to get maximum buy-in

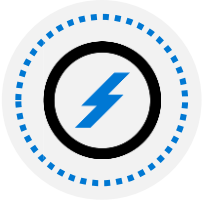
Identifying project metrics and Key Performance Indicators (KPIs)



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



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



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



Culture – Employee morale and retention rates

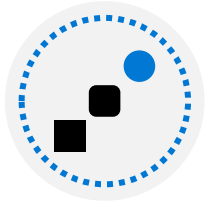


Goals must be specific, measurable, and time-bound

Lesson 04: Team structures

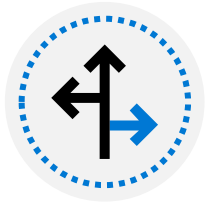


Agile development practices defined



Waterfall approach:

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



Agile approach:

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

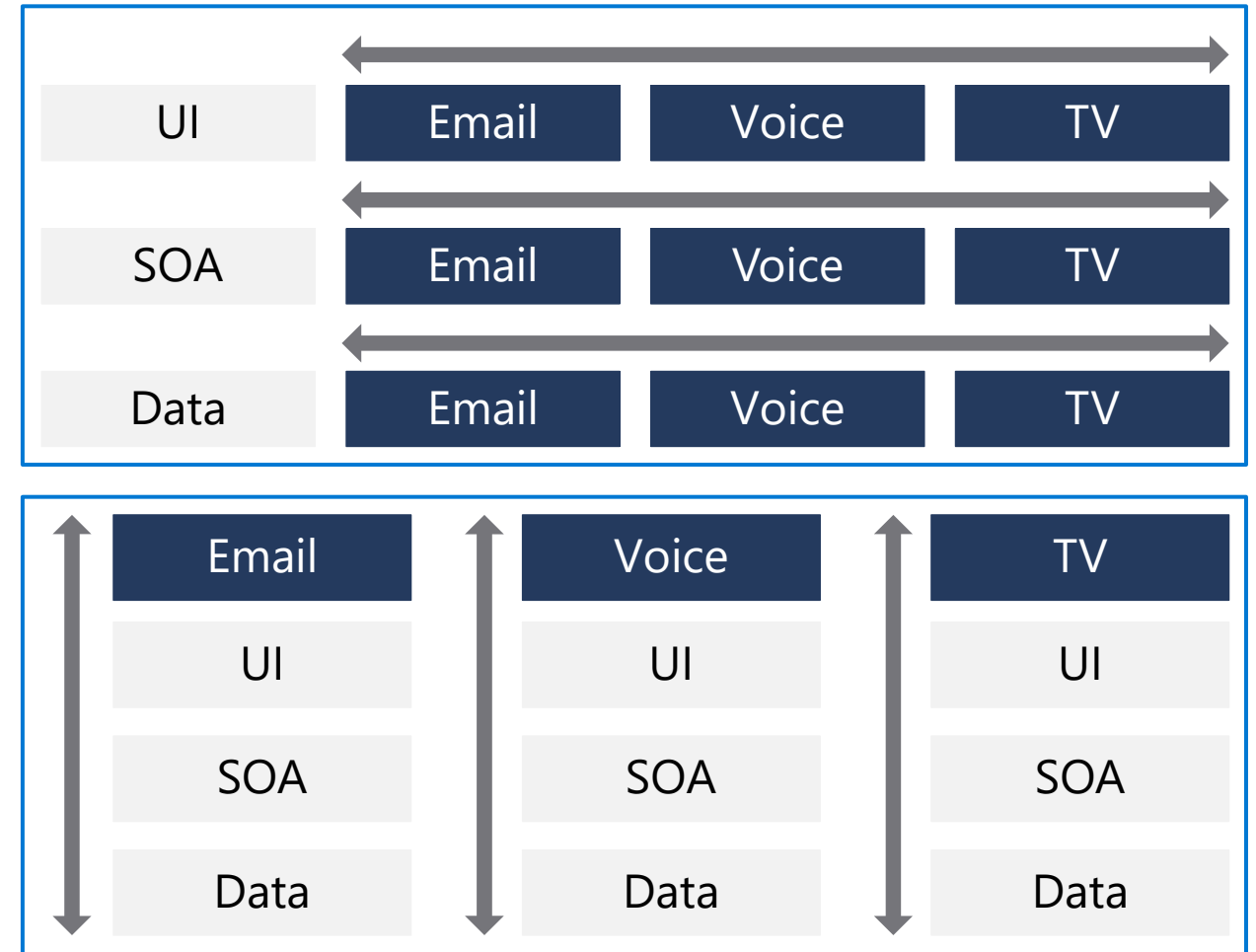
Principles of agile development

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

Creating organizational structures for agile practices

Horizontal team structures divide teams according to the software architecture.

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



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

Ideal DevOps team members



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

(#mw) also, see [T-People](#)

Mentoring team members on agile practices



Many teams hire external agile coaches or mentors



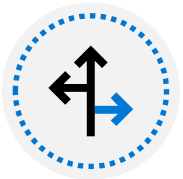
Agile coaches have teaching and mentoring skills



Agile coaches tend to be both trainers and consultants



Some coaches are technical experts



Some coaches are focused on agile processes



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

Enabling in-team and cross-team collaboration



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

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

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

Selecting tools and processes for agile practices



Tools can often enhance the outcomes achieved.

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

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

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

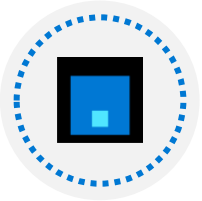
Lesson 05: Migrating to DevOps



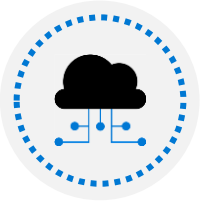
What can Azure DevOps do?



Azure Boards: Agile planning, work item tracking, visualization and reporting tool



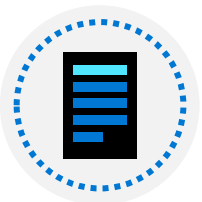
Azure Pipelines: A language, platform and cloud agnostic CI/CD platform with support for containers or Kubernetes



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

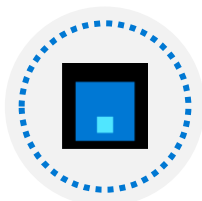


Azure Test Plans: Provides an integrated planned and exploratory testing solution

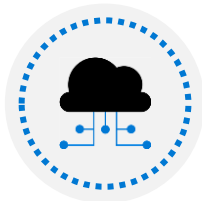
What can GitHub do?



Codespaces: Provide cloud-hosted collaborative development environments



Repos: Provide cloud-hosted and on-premises git repos for both public and private projects



Actions: Create automation workflows with environment variables and customized scripts

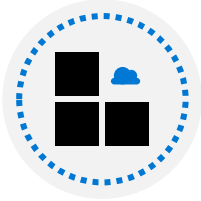


Artifacts: Ease integration with numerous existing packages and open-source repositories

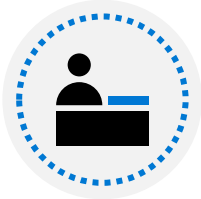


Security: Review code and identify vulnerabilities early in the development cycle

Designing an authorization and access strategy for Azure DevOps



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



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

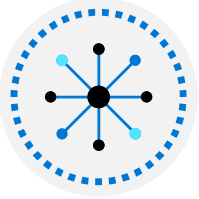


Azure DevOps is pre-configured with default security groups and permissions



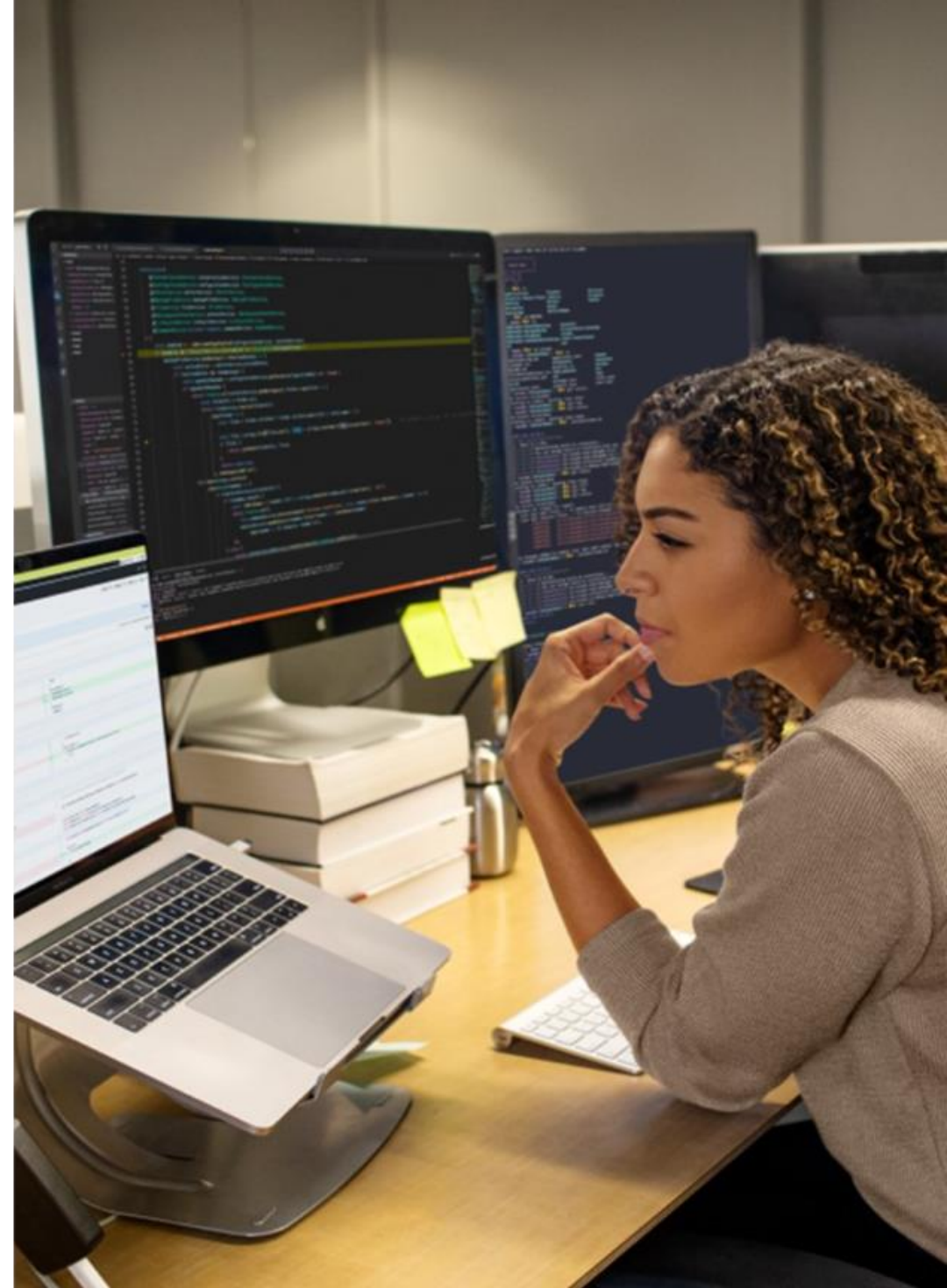
You can also configure app access policies and conditional access policies

Migrating or integrating existing work management tools

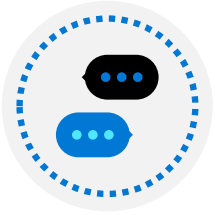


Both Azure DevOps and GitHub can be integrated with a wide variety of existing work management tools:

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



Migrating or integrating existing test management tools



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



Other tools:

[Apache JMeter](#) is open-source software written in Java and designed to load test functional behavior and measure performance.

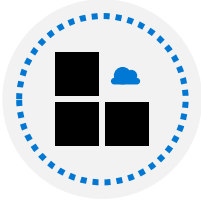
[Pester](#) is a tool that can be used to automate the testing of PowerShell code.

[SoapUI](#) provides another testing framework for SOAP and REST testing.



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

Designing a license management strategy



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

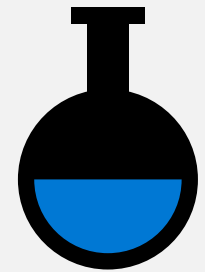
<https://azure.microsoft.com/en-us/pricing/details/devops/azure-devops-services/>



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

<https://github.com/pricing>

Lesson 06: Lab



Lab: Agile planning and portfolio management with Azure Boards

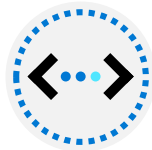


Lab overview:

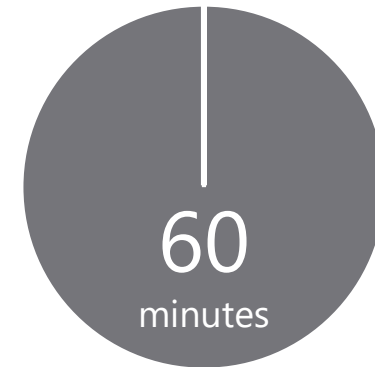
In this lab you will learn about the agile planning and portfolio management tools and processes provided by Azure Boards and how they can help you quickly plan, manage, and track work across your entire team.

Objectives:

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



Duration:



Lesson 07: Module review and takeaways



What did you learn?



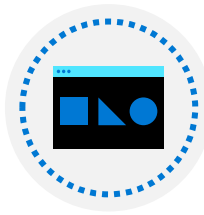
Plan for the transformation with shared goals and timelines



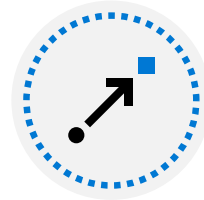
Select a project and identify project metrics and Key Performance Indicators (KPI's)



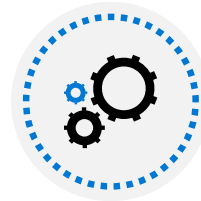
Create a team and agile organizational structure



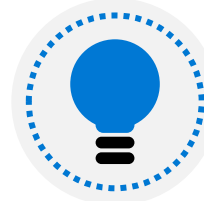
Design a tool integration strategy



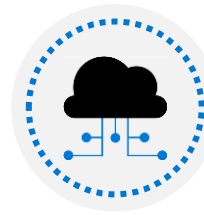
Design a license management strategy (e.g., Azure DevOps and GitHub users)



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

Module review questions

1

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

2

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

3

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

4

As a project metric, what is Lead Time measuring?

5

What is a cross-functional team?