Microsoft

PowerShell Advanced Workshop

GIT BASICS

Microsoft

Git design

Scenario overview

You are tired of having different versions of the same scripts running in your environment. You want to be able to leverage version control within your script repository.

Your engineers are confused on updates and which version of a specific script or file they are supposed to be used.

You want control over your scripts used in the production environment.

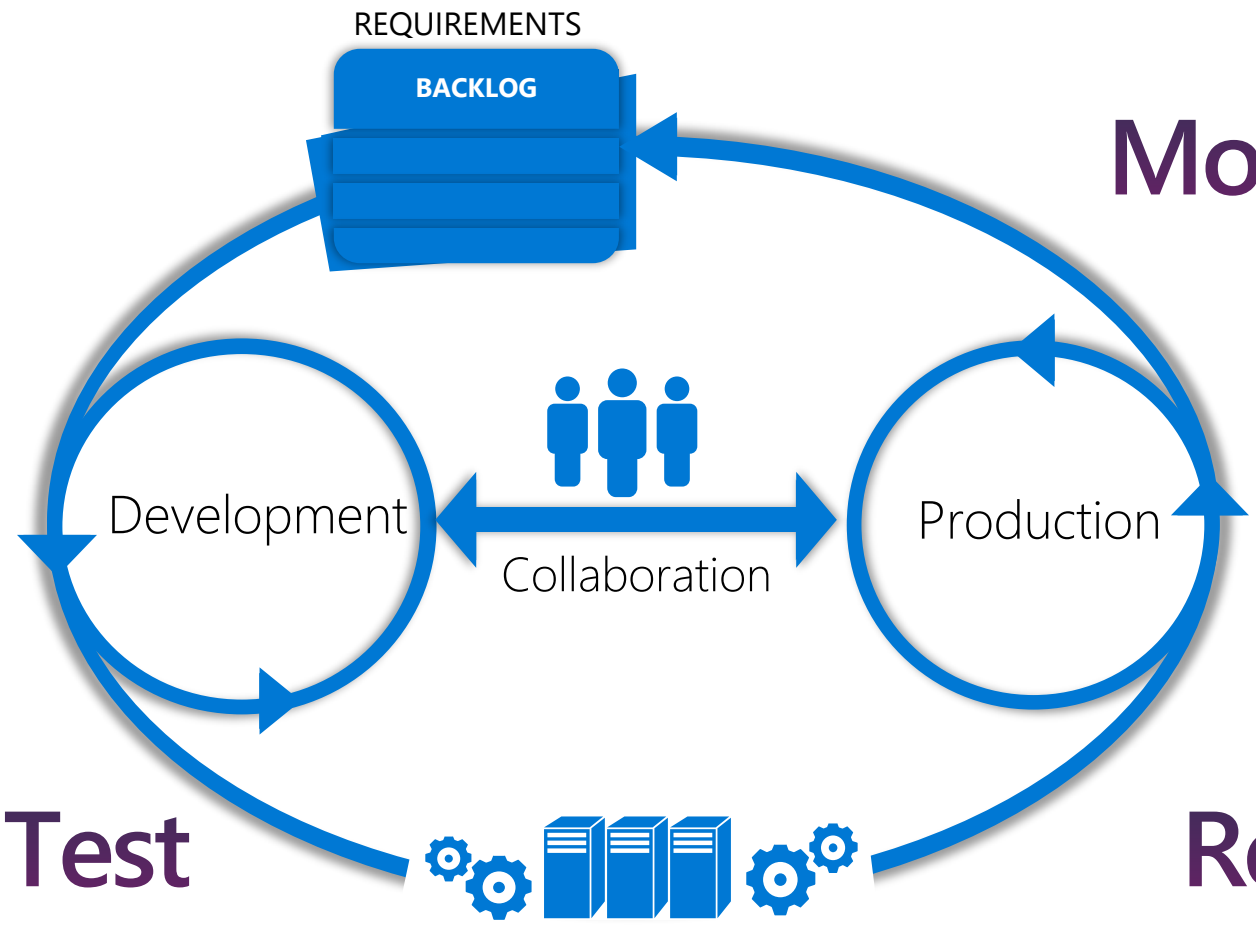
After completing this learning unit, you will be able to:

- Understand basics of DevOps and versioning processes.
- Understanding Git and GitHub options.

What is DevOps

- The union of people, process, and products to enable continuous delivery of value to your end users.
- The contraction of “Dev” and “Ops” refers to replacing siloed Development and Operations to create multidisciplinary teams that work together with shared and efficient practices and tools.

DevOps – deliver faster, smarter, and continuously



Introduction to version / source control

- Version control of scripts, files, applications, etc.
- Records changes to a file or set of files over time
- Allows file / project versioning
- Local or centralized
- Distributed version control change repository option
- Easily swap versions
- Ideal for teams
- Undo changes
- Stored on each machine

Why versioning and source control

Distributed

- Everyone has a copy
- Changes made locally
- Merged with an online central copy
- Doesn't generally require connectivity

Centralized

- Master copy is stored centrally
- Requires network connectivity

CI/CD strategy

Continuous Integration (CI)

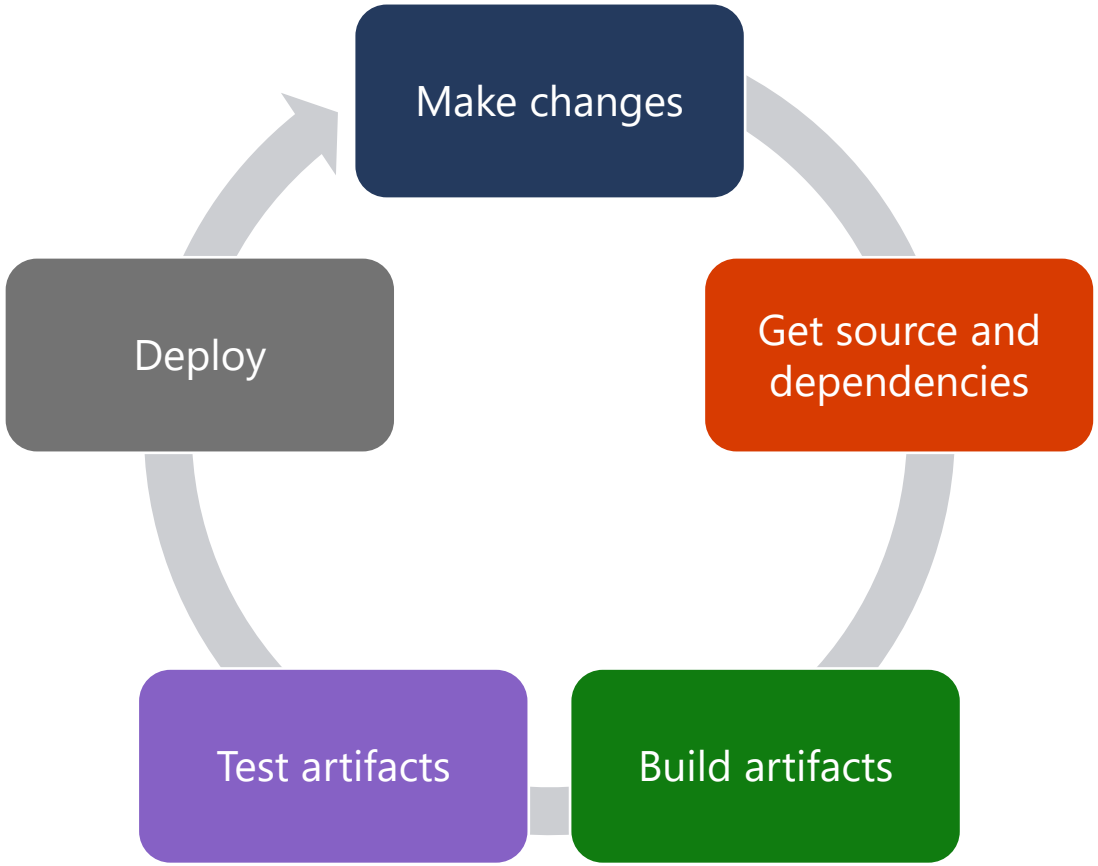
- Team develops from same repository
- Limited drift from master branch
- Results contain few bugs
- Software works properly

Continuous Delivery (CD)

- Produce software/updates in short cycles
- Allows: building, testing, and releasing code with greater speed and frequency
- Reduces: cost, time, and risk of delivering changes by allowing for more incremental updates to applications/scripts

Continuous innovation, continuous deployment

- Automation
- Repeatability
- Artifacts are:
  - internal PS modules
  - community PS modules
  - DSC resources
  - DSC configurations
  - .NET applications
  - WHATEVER YOU WANT



Source Control Management

- Harness collaboration
- Enable parallel development
- Minimize integration debt
- Act as a quality gate

Introduction to Git

- Distributed version control system
- Created in 2005 by Linus Torvalds
- Lightweight and fast
- Supports parallel development (branching)
- Fully distributed
- Open Source
- Integrated with many Content Index (CI) tools

What is Git

- New for many operational engineers
- Why afraid?
  - New toolset
  - Very powerful
  - Not user friendly
  - I don’t want to become a developer

Git changes

- Retains each version by tracking ALL changes
- Displays exact changes
- Undo is holistic and reverts

Git setup

- **Initialize**

- git init
- Adds .git folder

- **Clone**

- Create new repository to clone
- Copy existing to clone
- git clone ssh_url
- Adds .git folder
- Adds remote

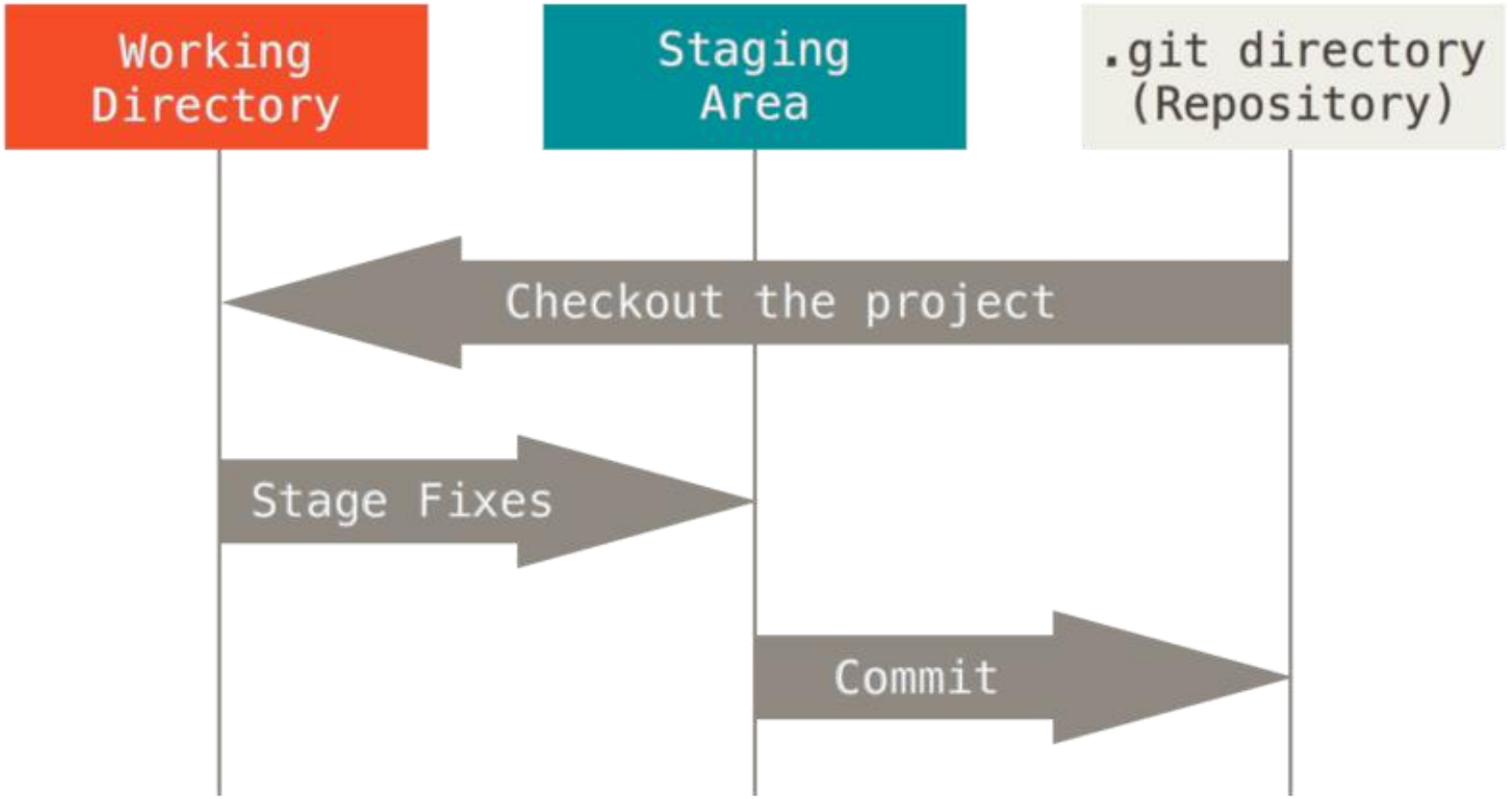
Git structure

- A Git repository is created by using **git init**
- Git creates a **.git** folder
- .**git** folder contains information needed for Git to function
- To remove Git, remove the **.git** folder and retain all project files

Git layers

Three stages to track changes:

- Working directory
- Index/staging
- HEAD



Microsoft

Logging into a Git repository

Repositories

- Can be either hosted online or local (GitHub, GitHub Enterprise)
- Represents the main project
- Contains: files, commits, branches, history for the entire project
- Best practices:
  - Maintain a single source repository
  - Place all needed content for a project in a single repository
  - Minimize branches

Git vs. GitHub

- Git: a revision control system, a tool to manage your source code history

- GitHub: a hosting service for Git repositories, only stores code

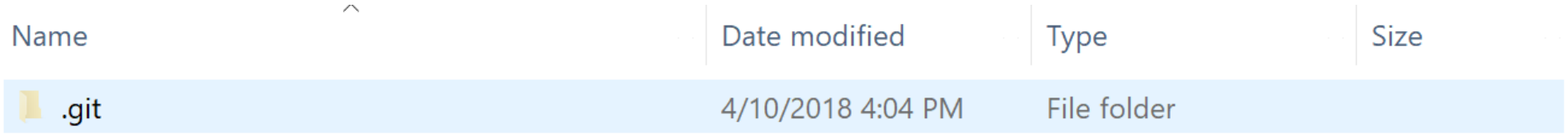
Create repository

Create New – git init <<RepositoryName>>

```
PS c:\> git init MyNewRepository
Initialized empty Git repository in C:/MyNewRepository/.git/
```

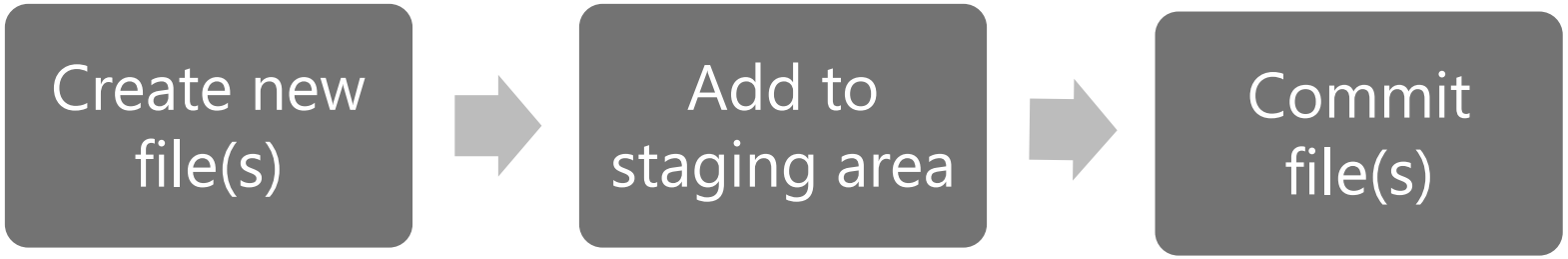
Clone from another location – git clone <<location .git>>

```
PS c:\> git clone https://github.com/anwaterh/MyNewRepository.git
Cloning into 'MyNewRepository'...
```



Workspace / staging area

- Files are created / edited in your workspace
- To track a file it first must be added to the staging area
- Staged files can then be committed (snapshot of the current staging area)
- All changes can be added – or select individual files



Adding content to staging area

- Add a file to staging area – git add <<filename>>

```
PS c:\> c:\MyNewRepository> git add .\script.txt
PS c:\> c:\MyNewRepository> git status
On branch master

No commits yet

Changes to be committed:
  (use “git rm –cached <file>...” to unstage)

          new file:     script.txt
```

- Add all files to staging area – git add

Removing staging area content

- Remove a file from staging (won't be part of the commit)

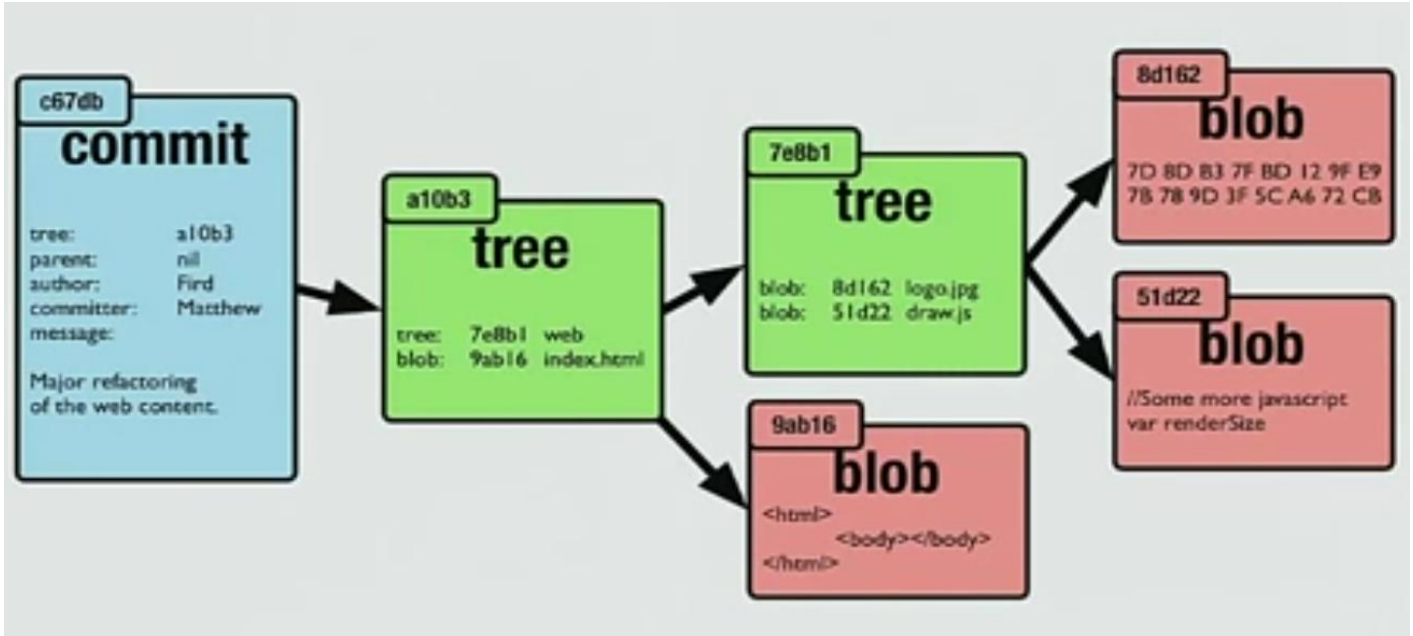
```
PS c:\> c:\MyNewRepository> git rm .\script.txt –cached
Rm 'script.txt'
PS c:\> c:\MyNewRepository> git status
On branch master

No commits yet

Untracked files:
   (use "git add <file>..." to include in what will be committed)

           script.txt

Nothing added to commit but untracked files present (use "git add" to track)
```

Commit

- Snapshots the repository at that point
- All staged files become part of the commit
- Saves email address and user name
- Contains a message describing the changes
- Commit has a unique hash which is used as a reference
- Use a text editor or the **–m** switch to provide the message

Commit under the hood

- Commit stores the entire blob, not just deltas as hashes: 530a6ac7c3682458ec6307a7c2c350ed849bfaf7
- Not duplicating content – leverage same blob
- Multiple commits can point to the same blob content
- Commit points to a tree, then to a blob
- Object DB used
- Start with root tree
- Updates parent
- **.git** folder to start



Committing changes

- Commit all changes – git commit –m "<<commit message>>"

```
PS c:\MyNewRepository> git commit –m "Initial commit"
[master (root-commit) 5619b86] Initial commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 script.txt
PS c:\MyNewRepository>
```

- Use the –amend switch to change the last commit

```
PS c:\MyNewRepository> git commit –m "Altered commit" --amend
[master 67b2b30] Altered commit
 Date: Tue Apr 10 16:21:50 2018 +1000
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 script.txt
PS c:\MyNewRepository>
```

Viewing history

- **git log**

```
PS c:\MyNewRepository> git log
Commit ebf96ff9862bc450a4b61c48742c0ec64639d1c (HEAD -> master)
Author: Anthony Watherston <anwather@contoso.com>
Date    Tue Apr 10 16:25:40 2018 +1000

    Made a small change

Commit 67b2b30e8cc6b8167ba947a55440730951ac44bq
Author: Anthony Watherston <anwather@contoso.com>
Date    Tue Apr 10 16:25:40 2018 +1000

     Altered commit
```

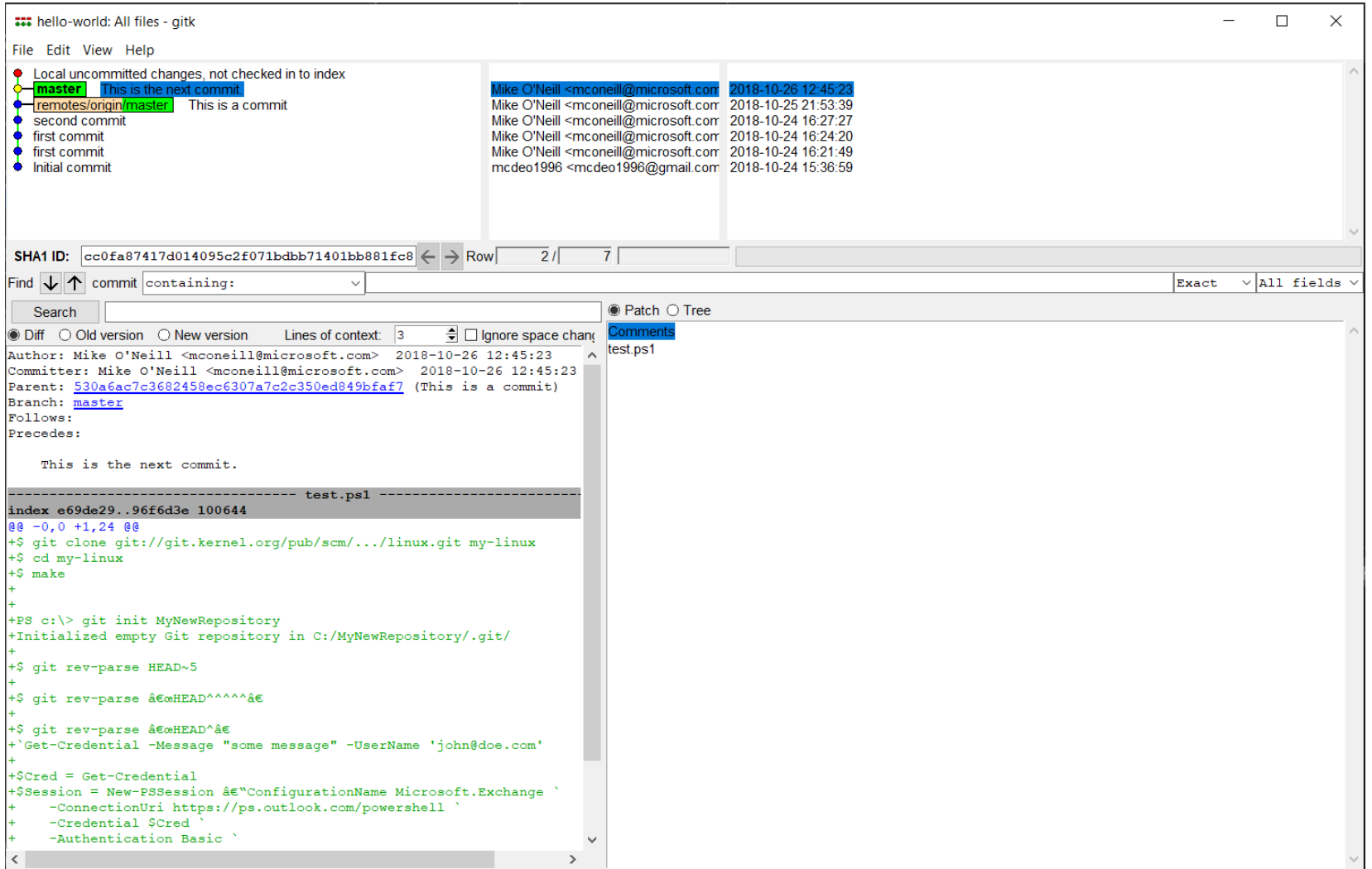
History options

- Many options to condense output and add decorations

```
PS c:\MyNewRepository> git log --oneline
ebf96ff (HEAD -> master) Made a small change
Commit 67b2b30 Altered commit
PS c:\MyNewRepository> git log –oneline --graph
* ebf96ff (HEAD -> master) Made a small change
* Commit 67b2b30 Altered commit
PS c:\MyNewRepository>
```

History options cont.

- Use **gitk** for a GUI history



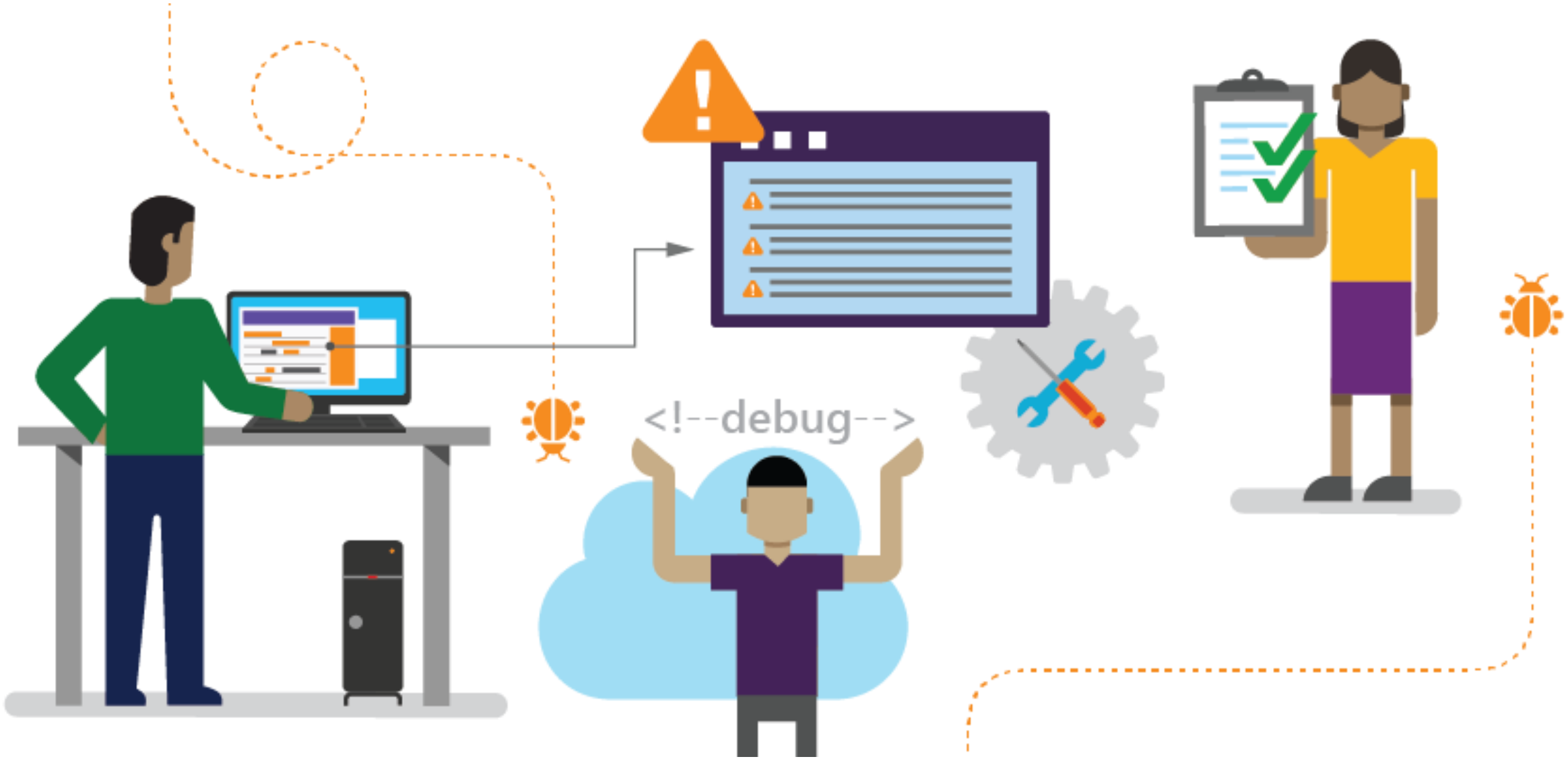
Demonstration

Logging into a Git repository

Microsoft

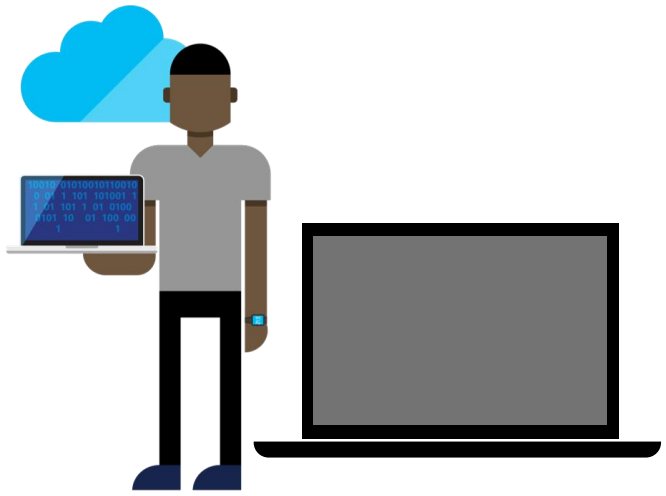
Branch and merging options

Development Struggles



Typical Lifecycle

Source Repository

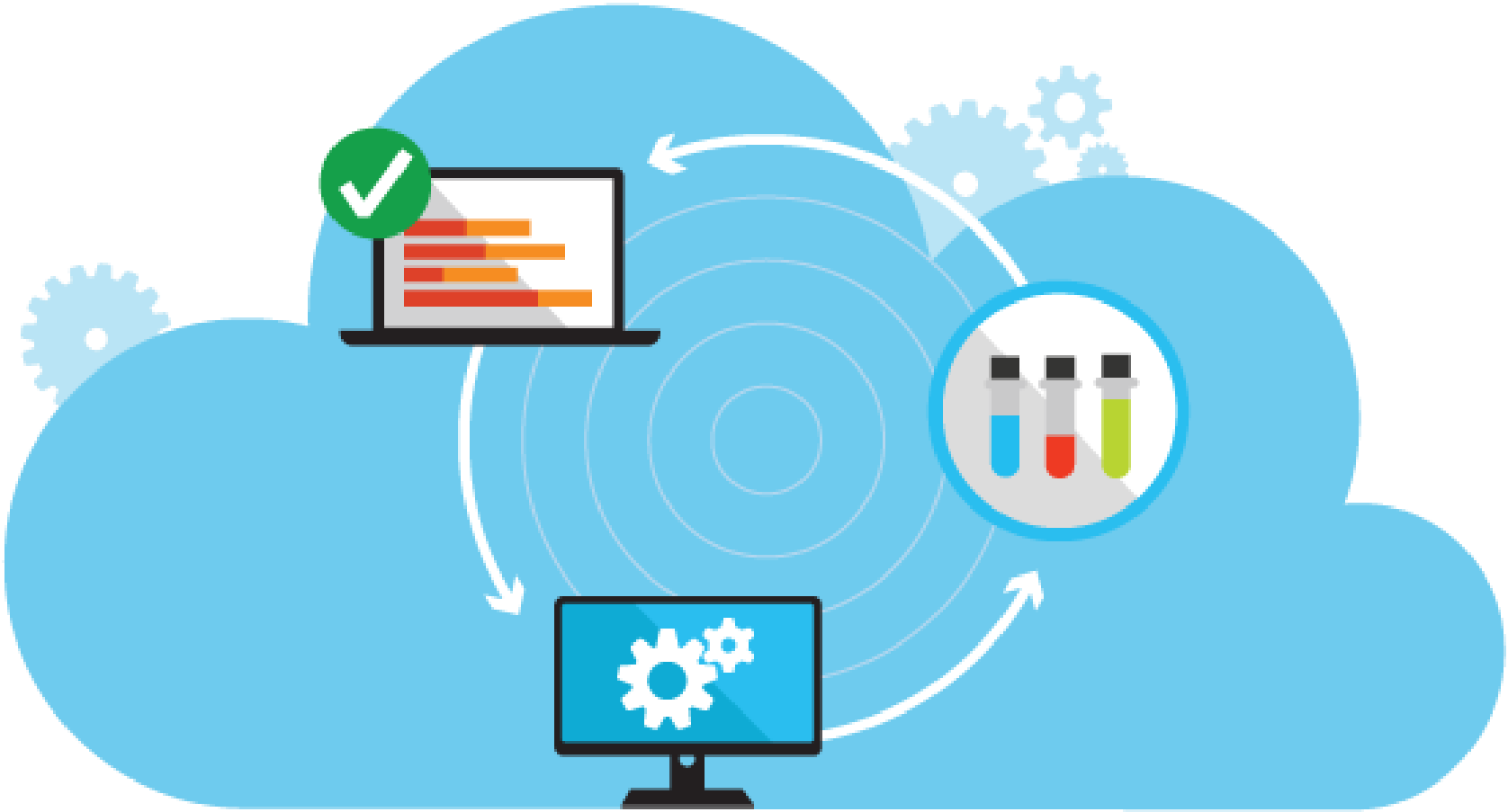


Build

The QA Environment

Production

Continuous Integration



Importance of Continuous Integration

Continuous Integration should be the backbone of development process

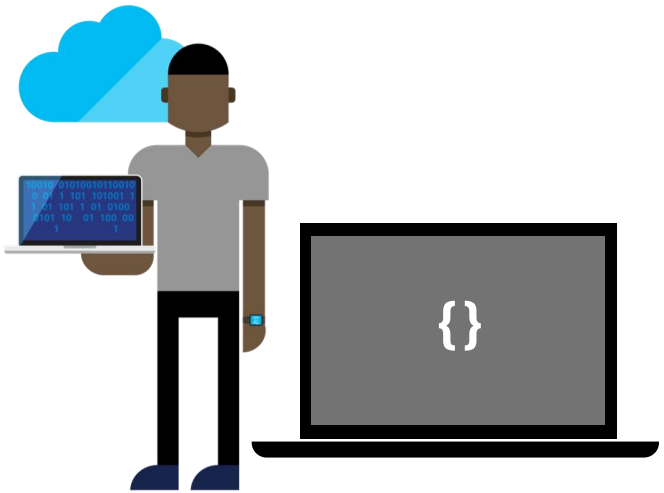
Continuous Integration results in:

- Improved quality
- Improved productivity
- Reduced risk

Continuous Integration – How It Works

Source Repository

Build

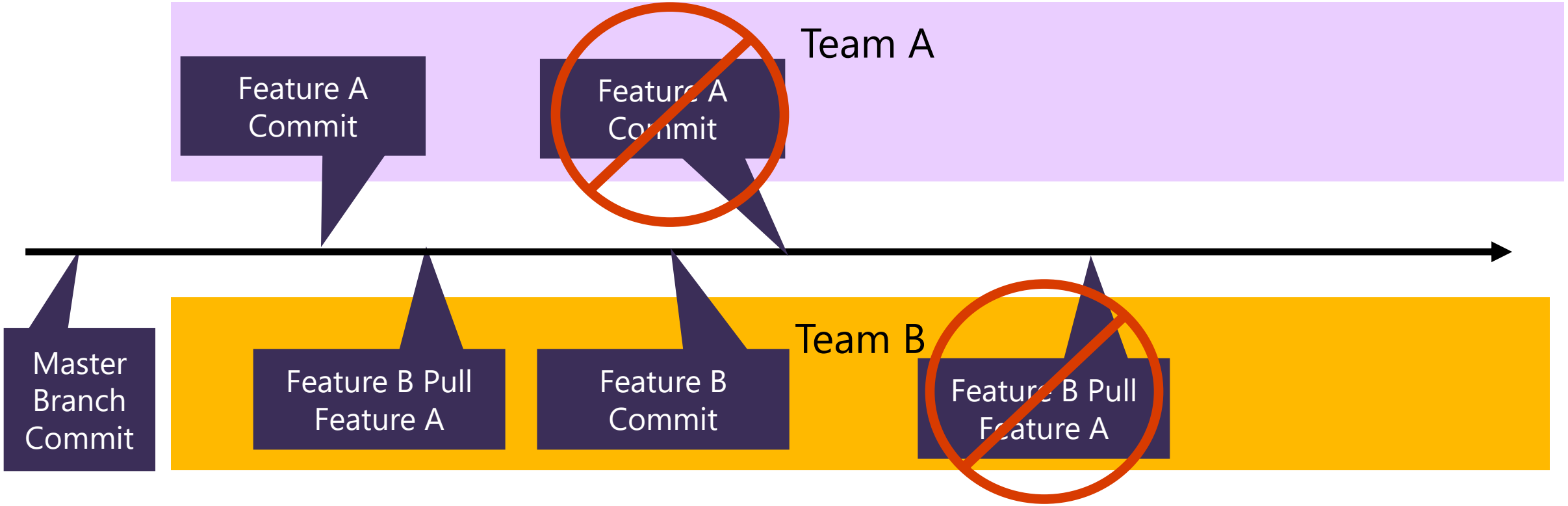


Test

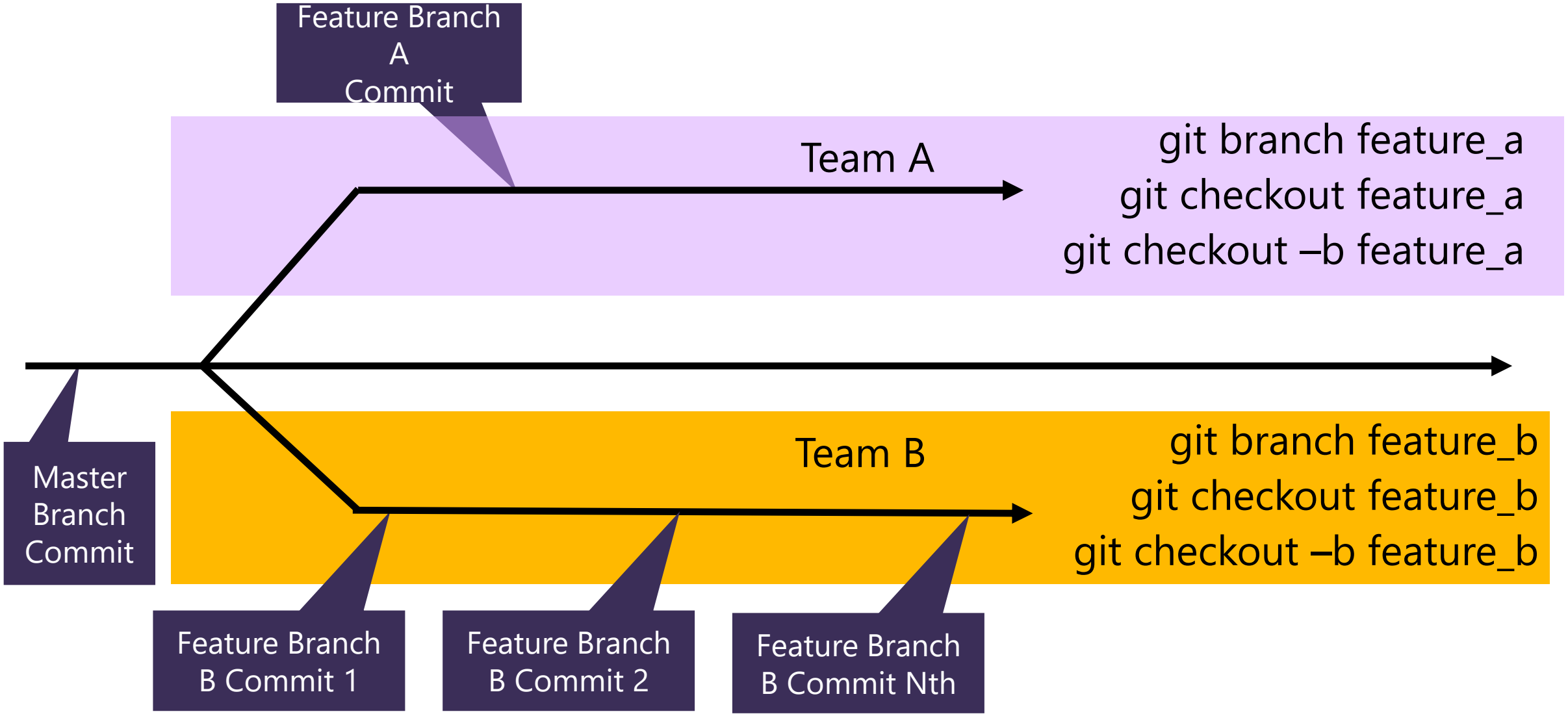
Branching high level

- Work can be done in branches
- Each branch is a separate line of development
- Does not affect any other branch
- Branches point to different commits
- Initial branch is called master

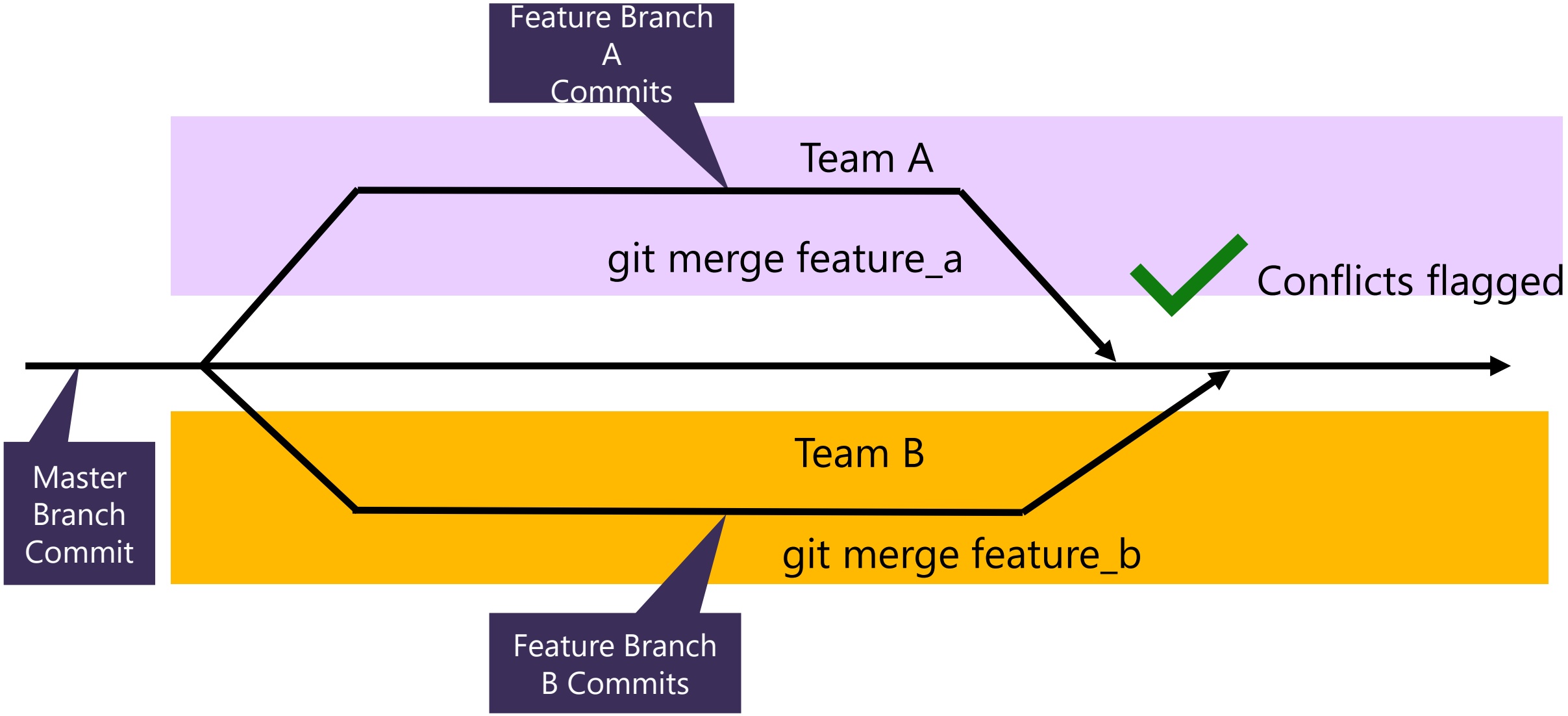
Why branching



Branching process



Branching solution - merge



Branch – HEAD

- Branches stored under the **refs/HEADS** tree
  - records tip-of-the-tree commit objects of branch name
  - **cat .git/HEAD**
  - Blobs stored as hashes: a113a6c7c3682458ec6307a7c2c350ed849bfaf7

Git status

- Displays paths that have differences between the index file and the current HEAD commit
- Paths that have differences between the working tree and the index file
- Paths in the working tree that are not tracked by Git

```
git status [<options>…] [--] [<pathspec>…]
```

New branch

- Add a new branch – git branch <<branch name>>
- git checkout <<branch name>>

```
PS c:\MyNewRepository> git branch dev
PS c:\MyNewRepository> git checkout dev
Switched to a new branch ‘dev’
```

- Create and switch to – git checkout –b <<branchname>>

```
PS c:\MyNewRepository> git checkout -b dev
Switched to a new branch ‘dev’
```

Viewing branches

- View all local branches – git branch

```
PS c:\MyNewRepository> git branch
* dev
  master
```

- View local and remote branches

```
PS c:\MyNewRepository> git branch --all
* dev
  master
  remotes/origin/master
```

Merging

- Brings together two branches
- Fast forward or recursive types
- Choose to keep or squash commits in topics branch
- Merge conflicts prevent changes being overwritten

Git merge

- git merge <<Branch Name>>

```
PS c:\MyNewRepository> git merge dev
Updating 5e9c4cb..b5cfbf5
Fast-forward
 file2.txt | 0
 1 file changed, 0 insertions(+), 0 deletions (-)
 create mode 100644 file2.txt
PS c:\MyNewRepository>
```

```
PS c:\MyNewRepository> git merge dev
Merge made by the ‘recursive’ strategy
 file2.txt | 1 +
 1 file changed, 1 insertions(+)
PS c:\MyNewRepository>
```