Azure DevOps

Hans-Petter Halvorsen

Contents

- Introduction to Azure DevOps
- Getting Started with Azure DevOps
  - Create and Configure a New Development Project
- Using **Agile/Scrum** Features in Azure DevOps
- **Source Code Control (SCC)** with Azure DevOps
- **Bug Tracking** with Azure DevOps

Introduction to Azure DevOps

Hans-Petter Halvorsen

Azure DevOps

- Tool for Software Engineering
  - Planning, Collaboration, Source Code Control, Bug Tracking, etc.
- Developed by Microsoft
- https://dev.azure.com
- Free for 5 Developers + Stakeholders

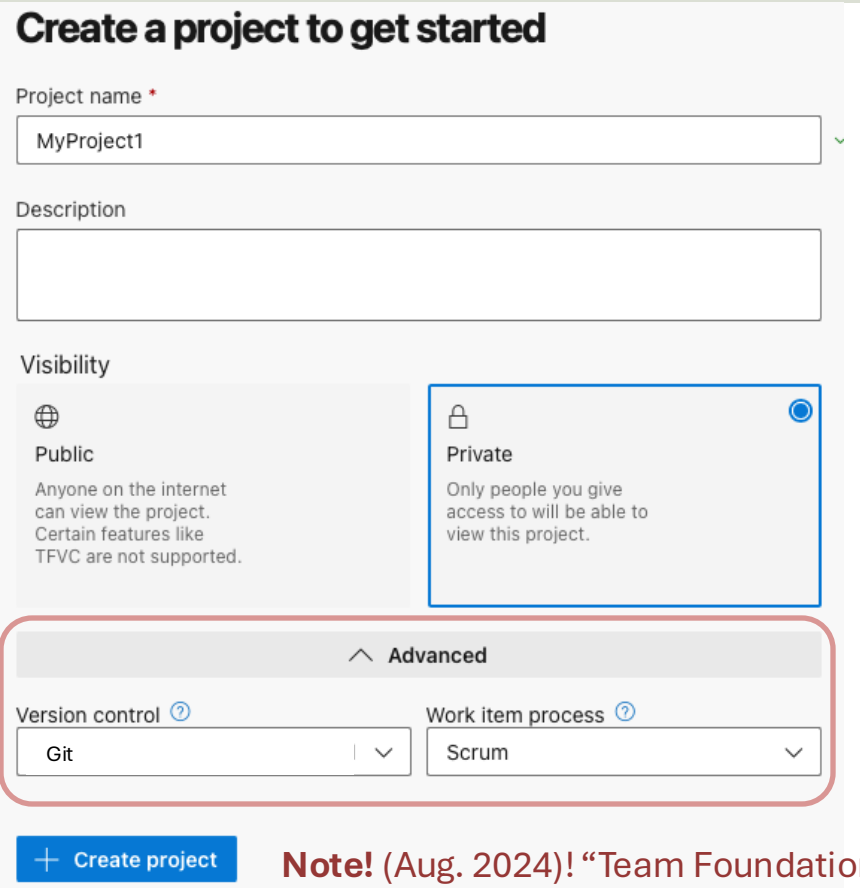
Features

- Integrated with Visual Studio
- Agile/Scrum Tools
  - Product Backlog
  - Sprint Backlog and Taskboard
- Source Code Control (SCC)
- Bug Reporting and Tracking
- ++ (Not covered in this tutorial)

Getting Started with Azure DevOps

Hans-Petter Halvorsen

Create New Project



https://dev.azure.com

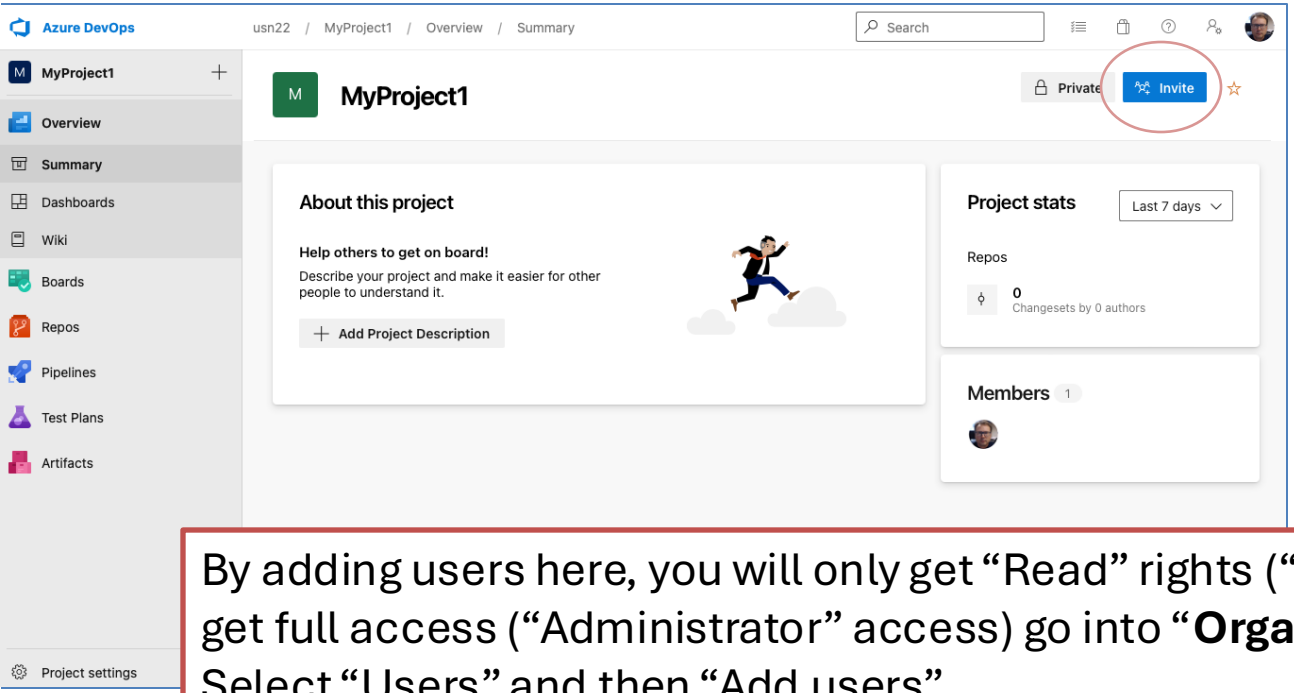
Enter a meaningful **Name** for your Project

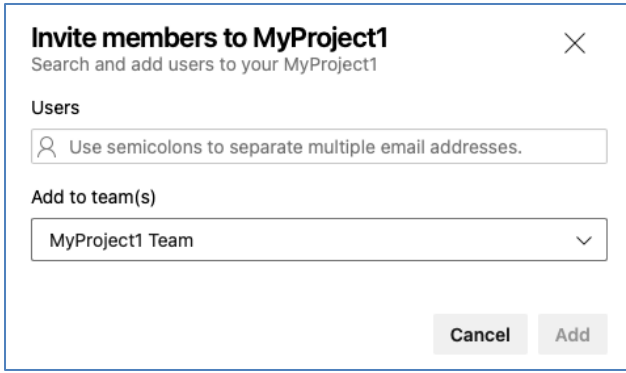
Enter a meaningful **Description** for your Project

Note!!!
Select "Advanced" Settings
Version control = **Git**
Work item process = **Scrum**

Note! (Aug. 2024)! "Team Foundation Version Control (TFVC)" is no longer supported for New Projects

Invite/Add Members

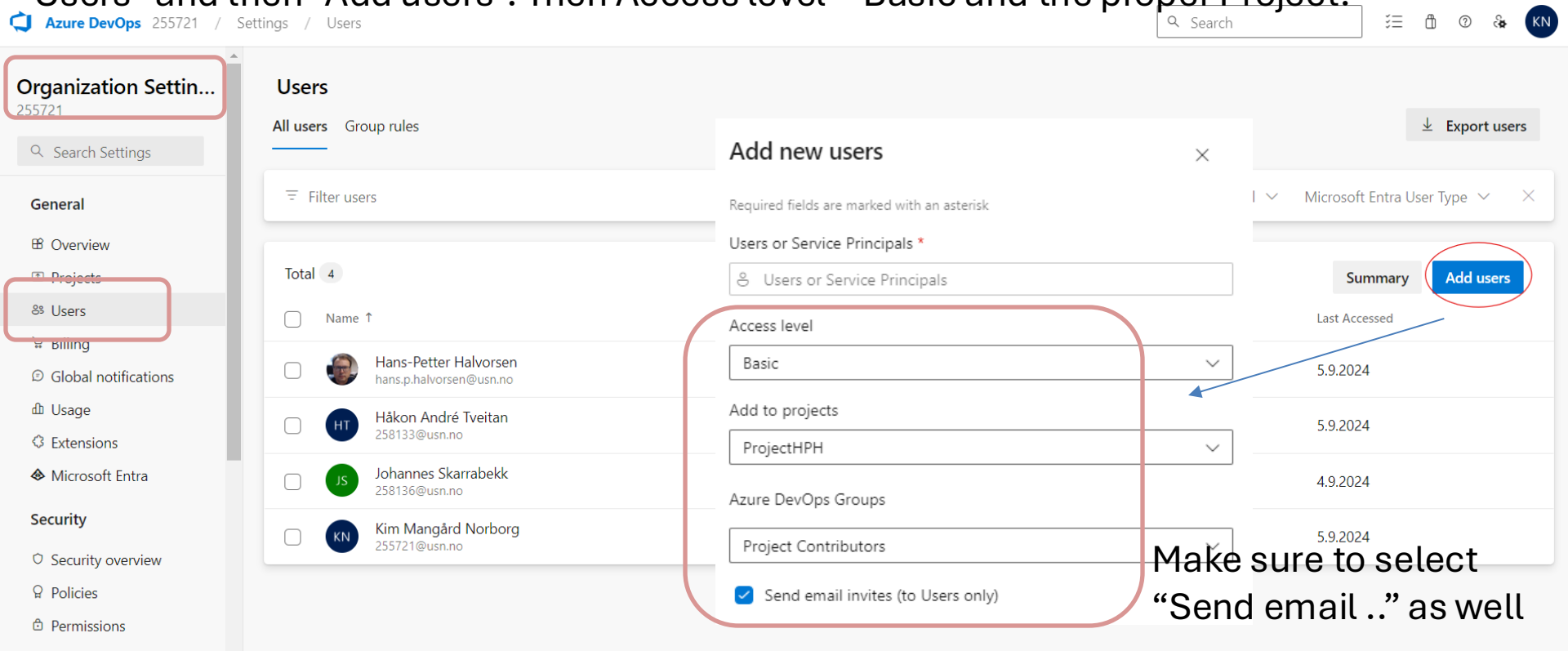




By adding users here, you will only get “Read” rights (“Stakeholder”). To get full access (“Administrator” access) go into “**Organization setting**”. Select “Users” and then “Add users”.

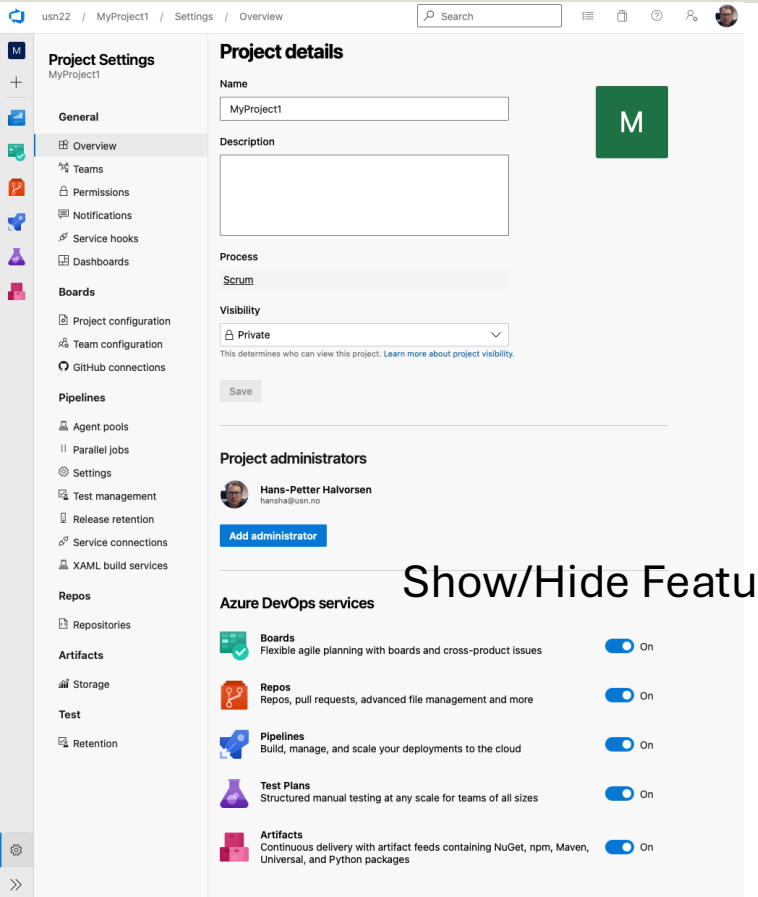
Give Users “Full Access” in Organization Settings

To get full access (“Administrator” access) go into “**Organization setting**”. Select “Users” and then “Add users”. Then Access level = Basic and the proper Project.

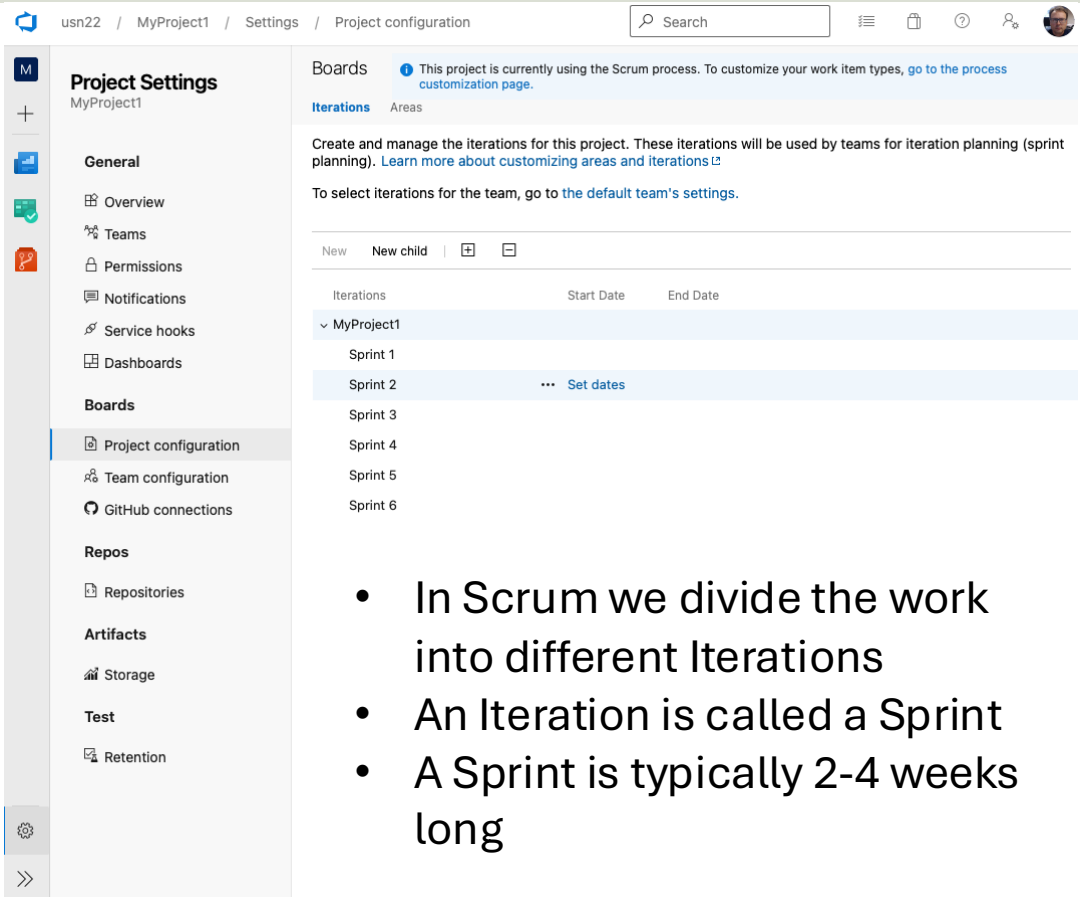


Make sure to select “Send email ..” as well

Project Settings



Show/Hide Features

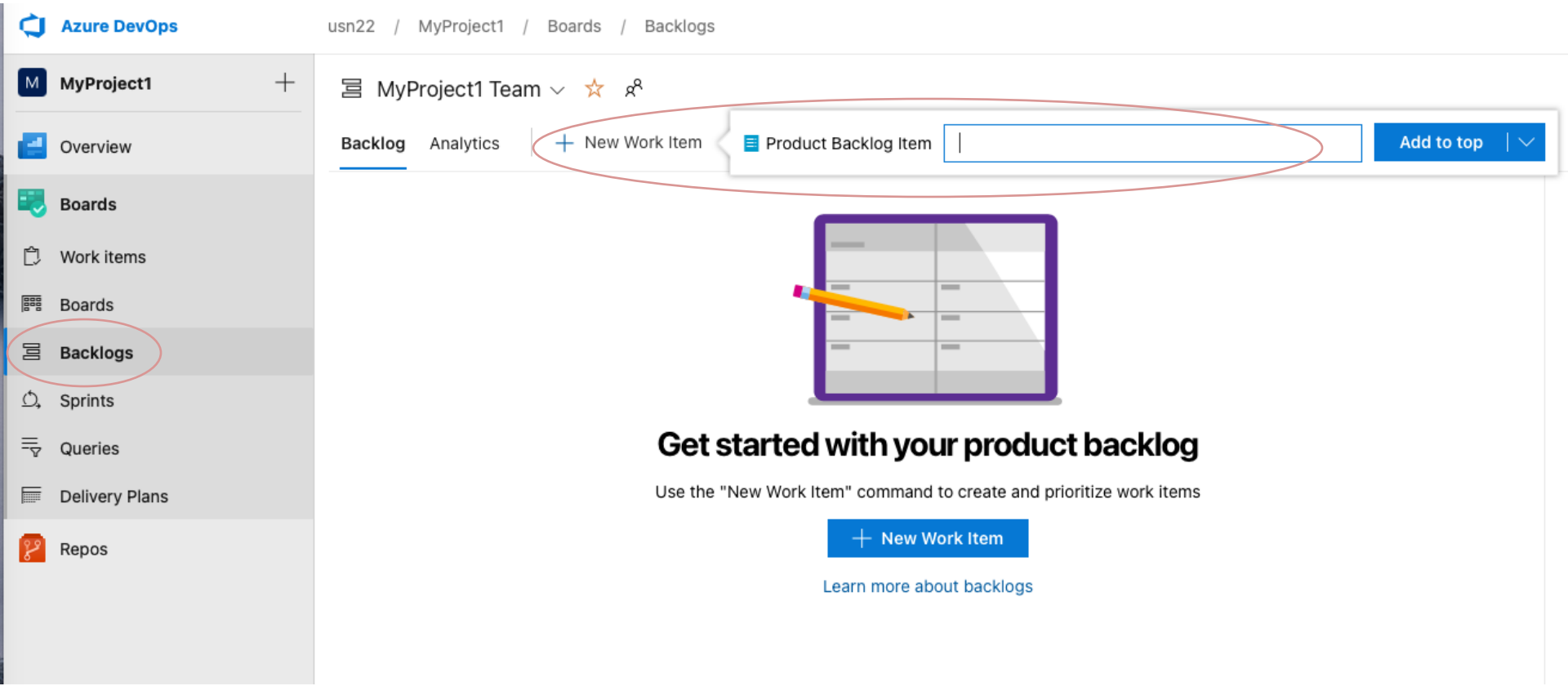


- In Scrum we divide the work into different Iterations
- An Iteration is called a Sprint
- A Sprint is typically 2-4 weeks long

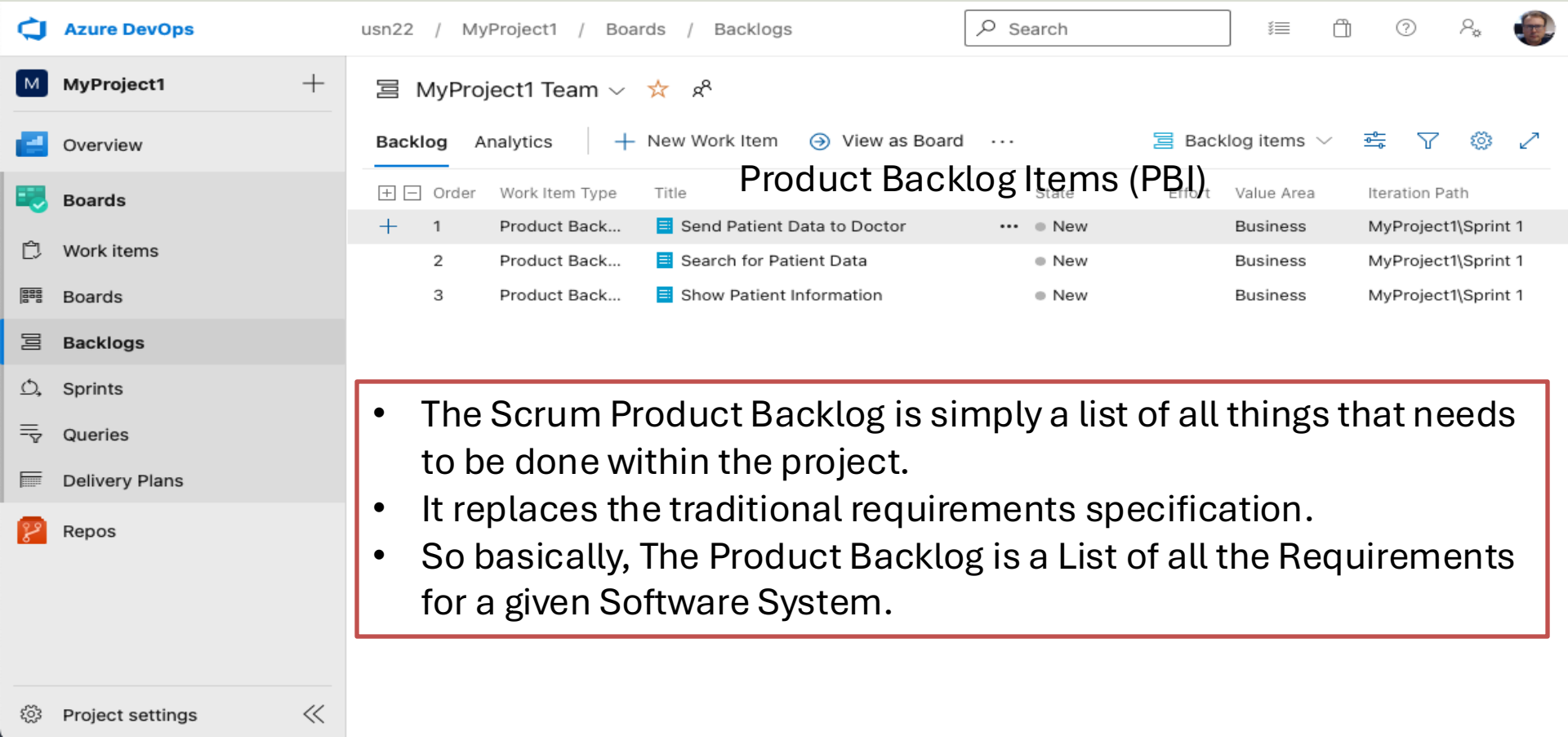
Agile/Scrum Features in Azure DevOps

Hans-Petter Halvorsen

Product Backlog

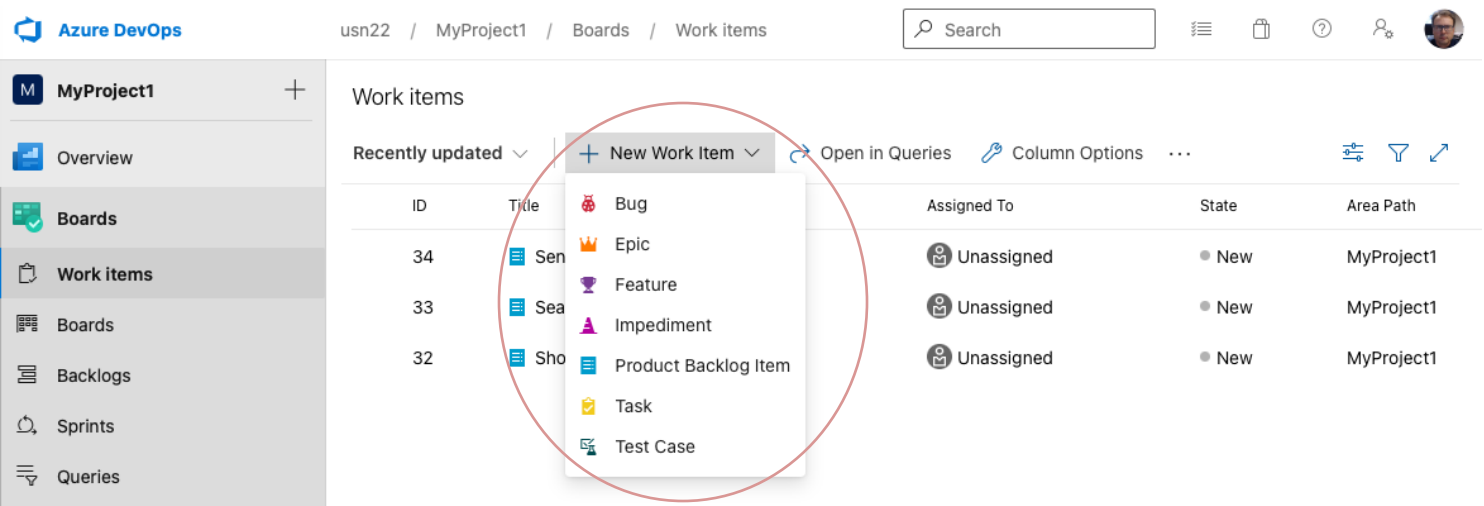


Product Backlog



- The Scrum Product Backlog is simply a list of all things that needs to be done within the project.
- It replaces the traditional requirements specification.
- So basically, The Product Backlog is a List of all the Requirements for a given Software System.

Work Items



3 important Work Item Types related to Product Backlog

- **Product Backlog Item** (PBI)
- **Feature** – Can be used to group PBIs that belong together
- **Task** – Divide a PBI into doable Tasks. Each PBI needs to be broken down into a set of Tasks. A Task is something that should be done by the developer

Feature → PBI → Task

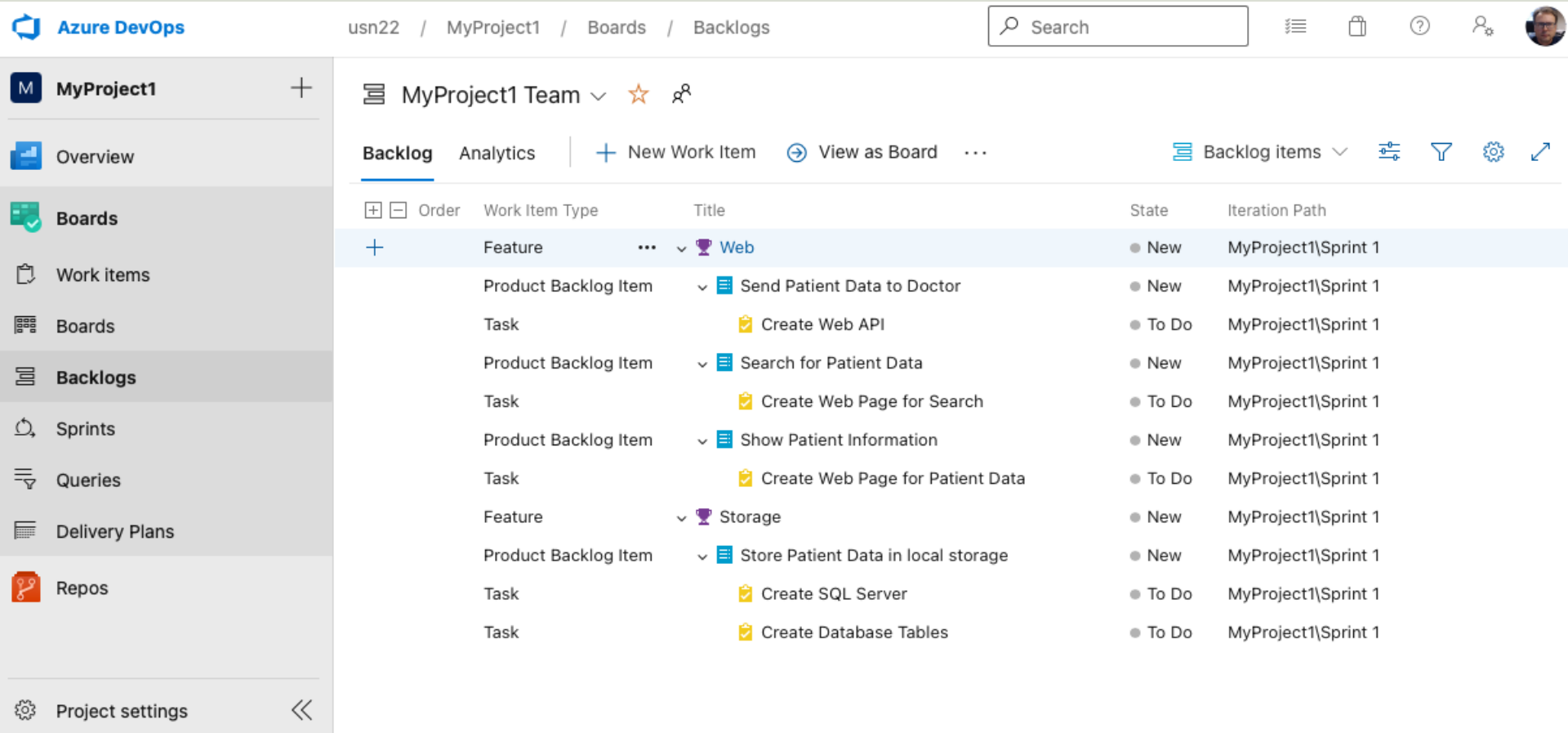
- Feature X
  - Product Backlog Item A
    - ✓ Task A.1
    - ✓ Task A.2
  - Product Backlog Item B
    - ✓ Task B.1
    - ✓ Task B.2
- Feature Y
  - Product Backlog Item C
    - ✓ Task C.1
    - ✓ Task C.2

etc.

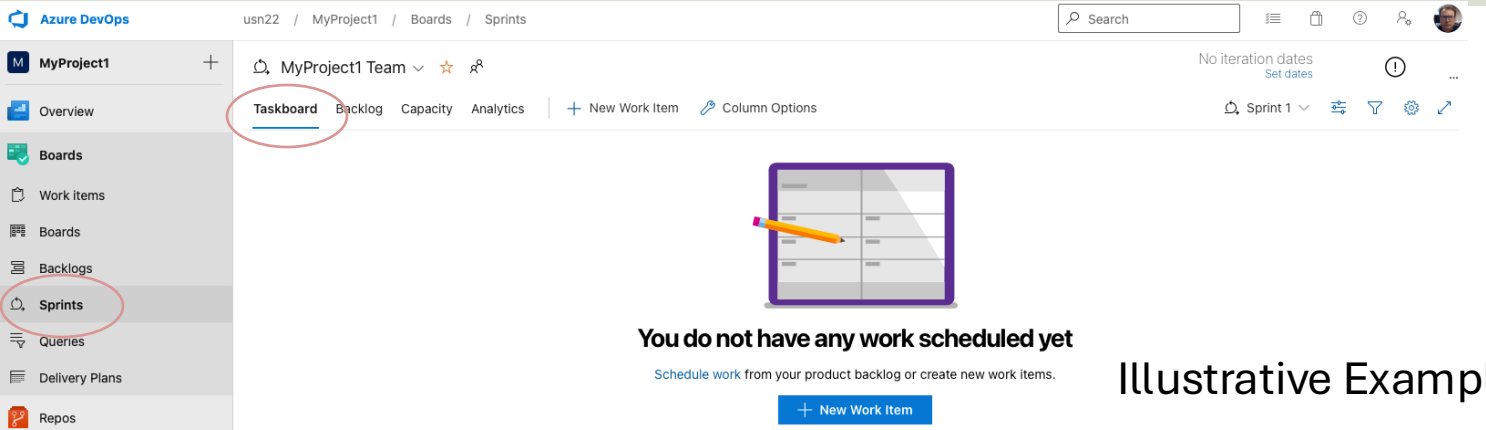
Features makes in easier to structure all the Product Backlog Items

Each Product Backlog Item is broken down in one or more Tasks by the Developer

Example


Azure DevOps
usn22 / MyProject1 / Boards / Backlogs
Search
MyProject1
Overview
Boards
Work items
Boards
Backlogs
Sprints
Queries
Delivery Plans
Repos
Project settings
MyProject1 Team
Backlog
Analytics
New Work Item
View as Board
Backlog items
Order
Work Item Type
Title
State
Iteration Path
Feature
Web
New
MyProject1\Sprint 1
Product Backlog Item
Send Patient Data to Doctor
New
MyProject1\Sprint 1
Task
Create Web API
To Do
MyProject1\Sprint 1
Product Backlog Item
Search for Patient Data
New
MyProject1\Sprint 1
Task
Create Web Page for Search
To Do
MyProject1\Sprint 1
Product Backlog Item
Show Patient Information
New
MyProject1\Sprint 1
Task
Create Web Page for Patient Data
To Do
MyProject1\Sprint 1
Feature
Storage
New
MyProject1\Sprint 1
Product Backlog Item
Store Patient Data in local storage
New
MyProject1\Sprint 1
Task
Create SQL Server
To Do
MyProject1\Sprint 1
Task
Create Database Tables
To Do
MyProject1\Sprint 1

Sprints/Taskboard

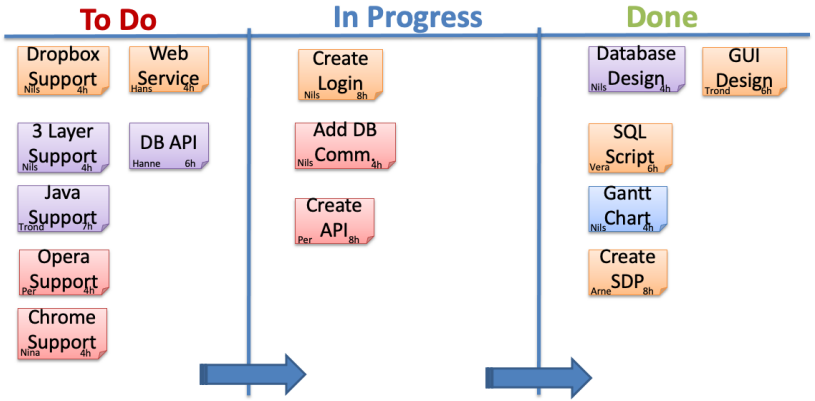


A Taskboard in Scrum has 3 columns

- **To Do**
- **In Progress**
- **Done**

Within a Sprint you move the Tasks from left to right as the status changes

Illustrative Example:



Sprints/Taskboard

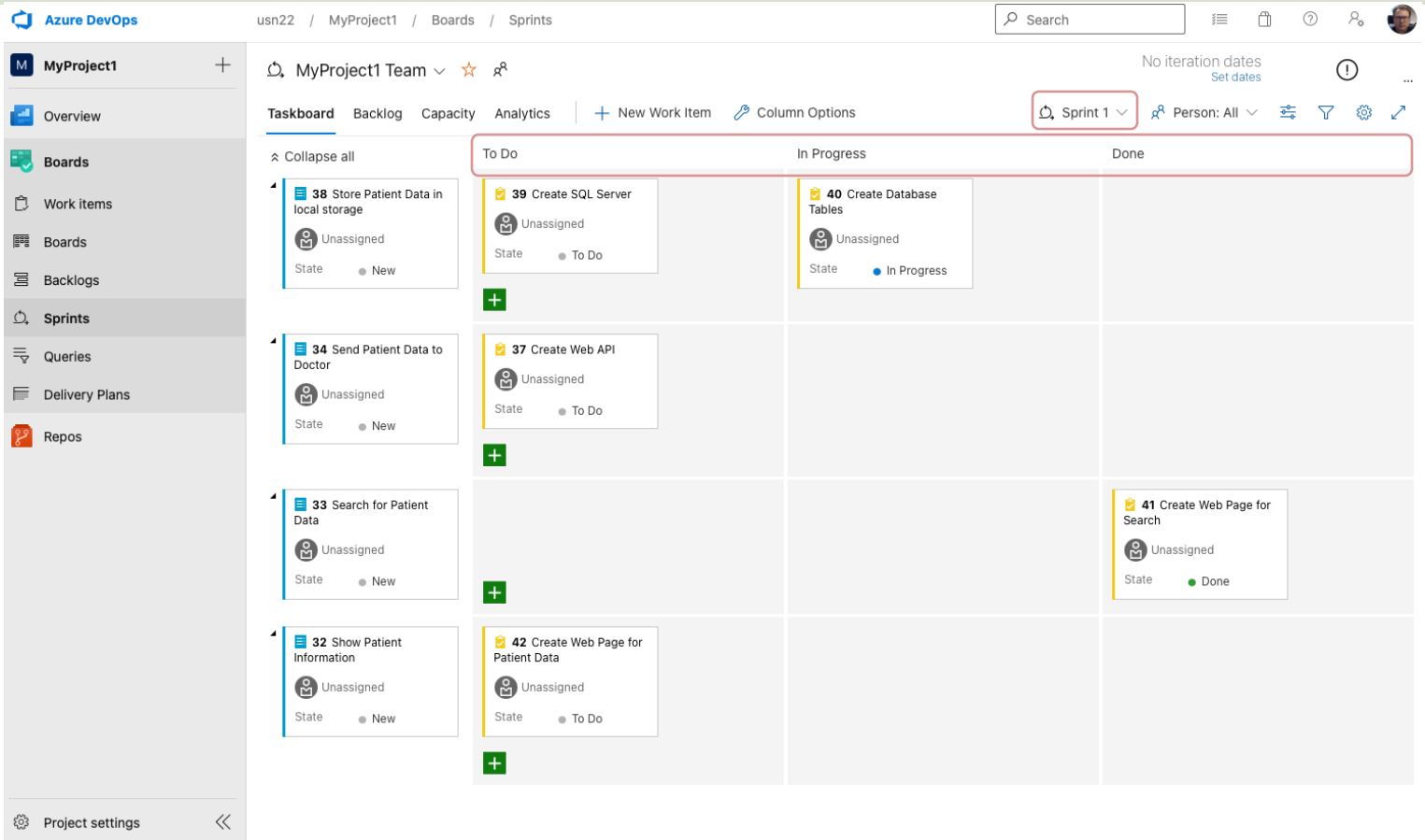
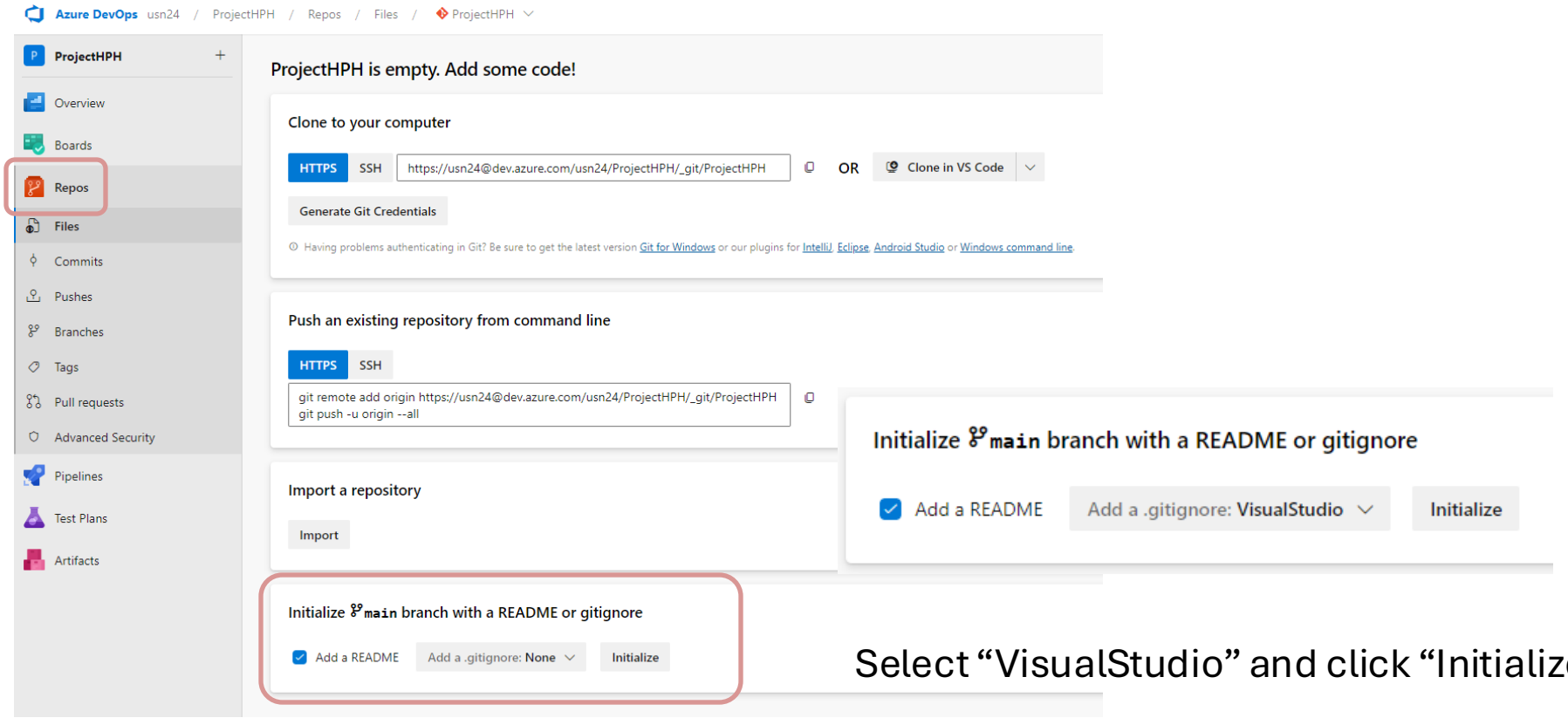


Table of Contents

Source Code Control with Azure DevOps

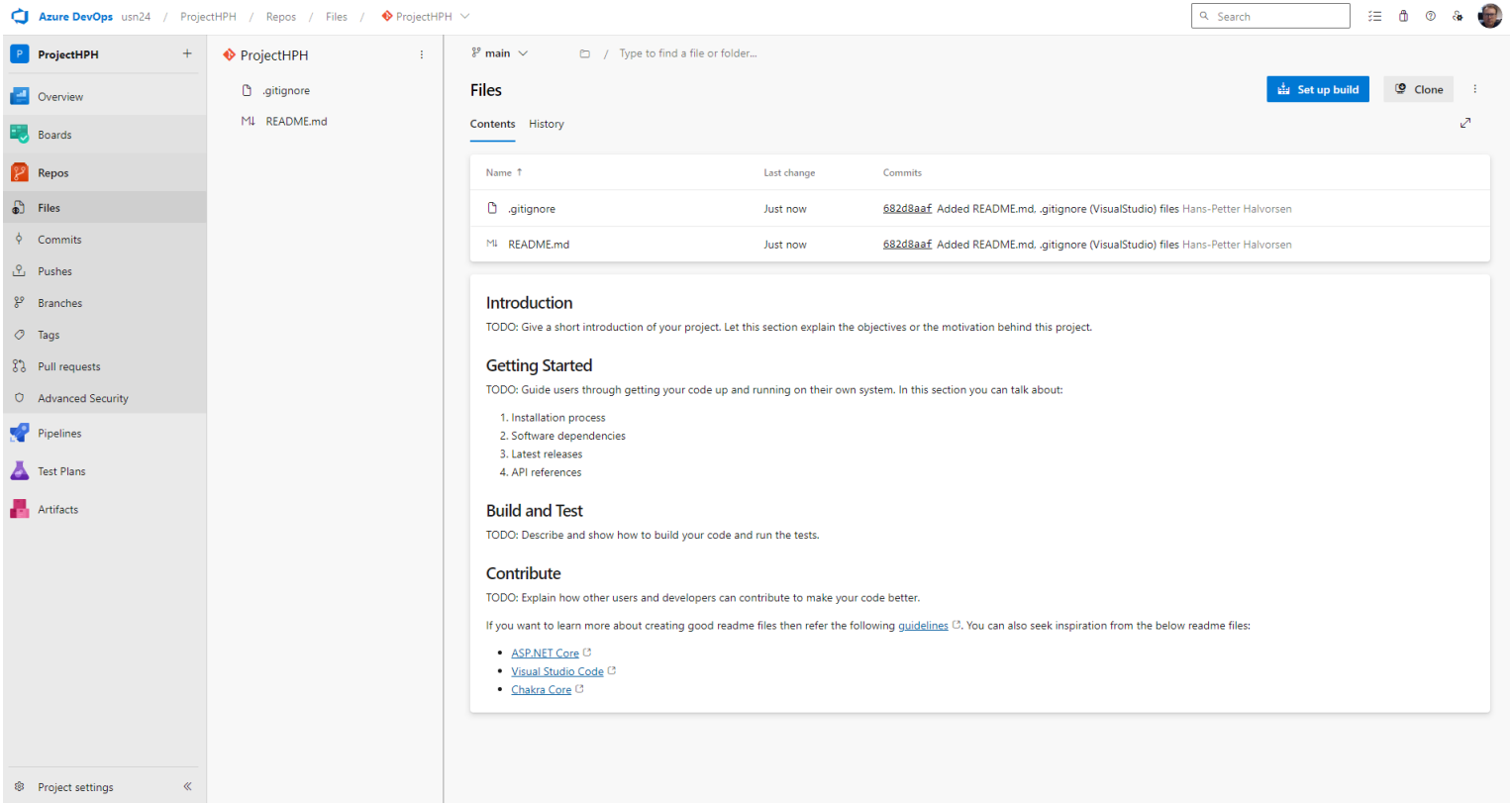
Hans-Petter Halvorsen

Configure **.gitignore** File



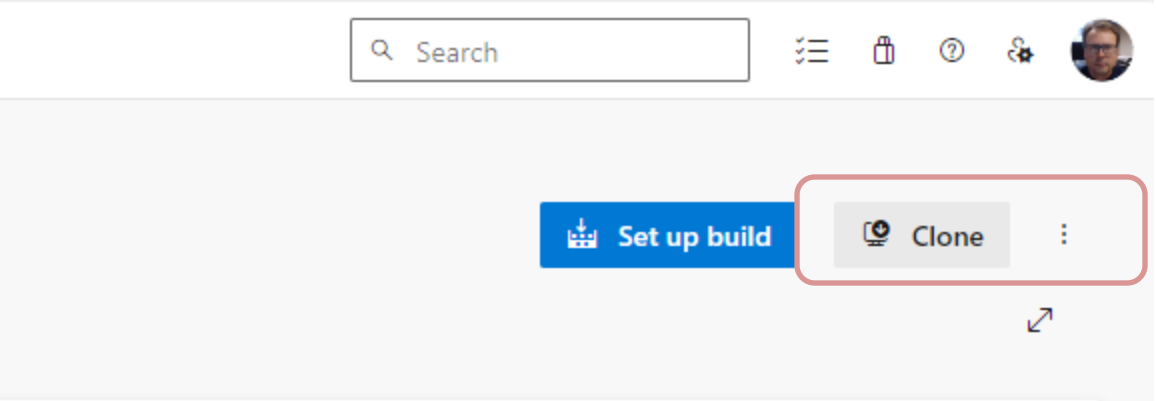
Select “VisualStudio” and click “Initialize”

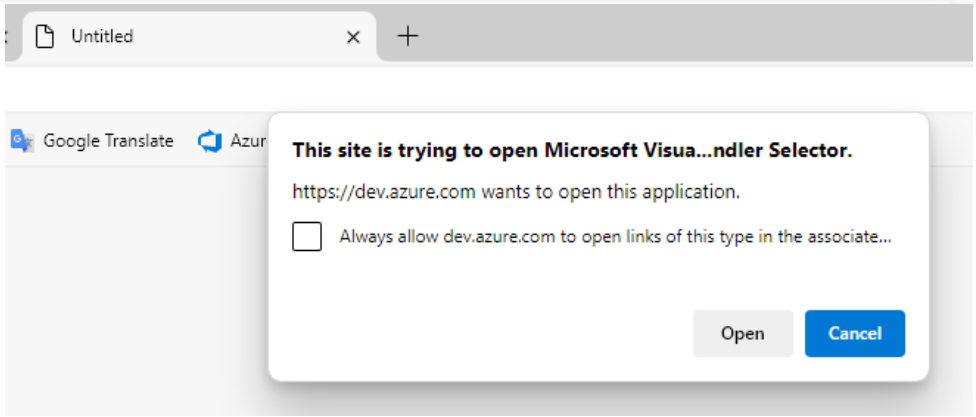
.gitignore

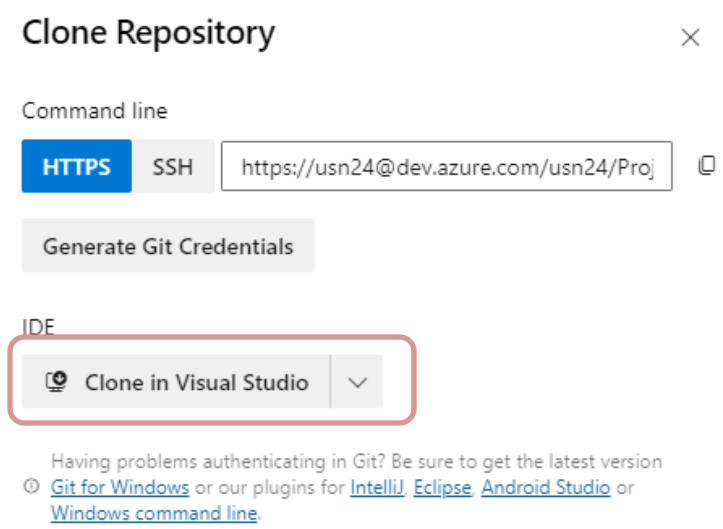

Azure DevOps usn24 / ProjectHPH / Repos / Files / ProjectHPH
Search
ProjectHPH
Overview
Boards
Repos
Files
Commits
Pushes
Branches
Tags
Pull requests
Advanced Security
Pipelines
Test Plans
Artifacts
Project settings
ProjectHPH
.gitignore
README.md
main
Type to find a file or folder...
Files
Set up build
Clone
Contents
History
Name
Last change
Commits
.gitignore
Just now
682d8aaf Added README.md, .gitignore (VisualStudio) files Hans-Petter Halvorsen
README.md
Just now
682d8aaf Added README.md, .gitignore (VisualStudio) files Hans-Petter Halvorsen
Introduction
TODO: Give a short introduction of your project. Let this section explain the objectives or the motivation behind this project.
Getting Started
TODO: Guide users through getting your code up and running on their own system. In this section you can talk about:
1. Installation process
2. Software dependencies
3. Latest releases
4. API references
Build and Test
TODO: Describe and show how to build your code and run the tests.
Contribute
TODO: Explain how other users and developers can contribute to make your code better.
If you want to learn more about creating good readme files then refer the following guidelines. You can also seek inspiration from the below readme files:
ASP.NET Core
Visual Studio Code
Chakra Core

Local Clone

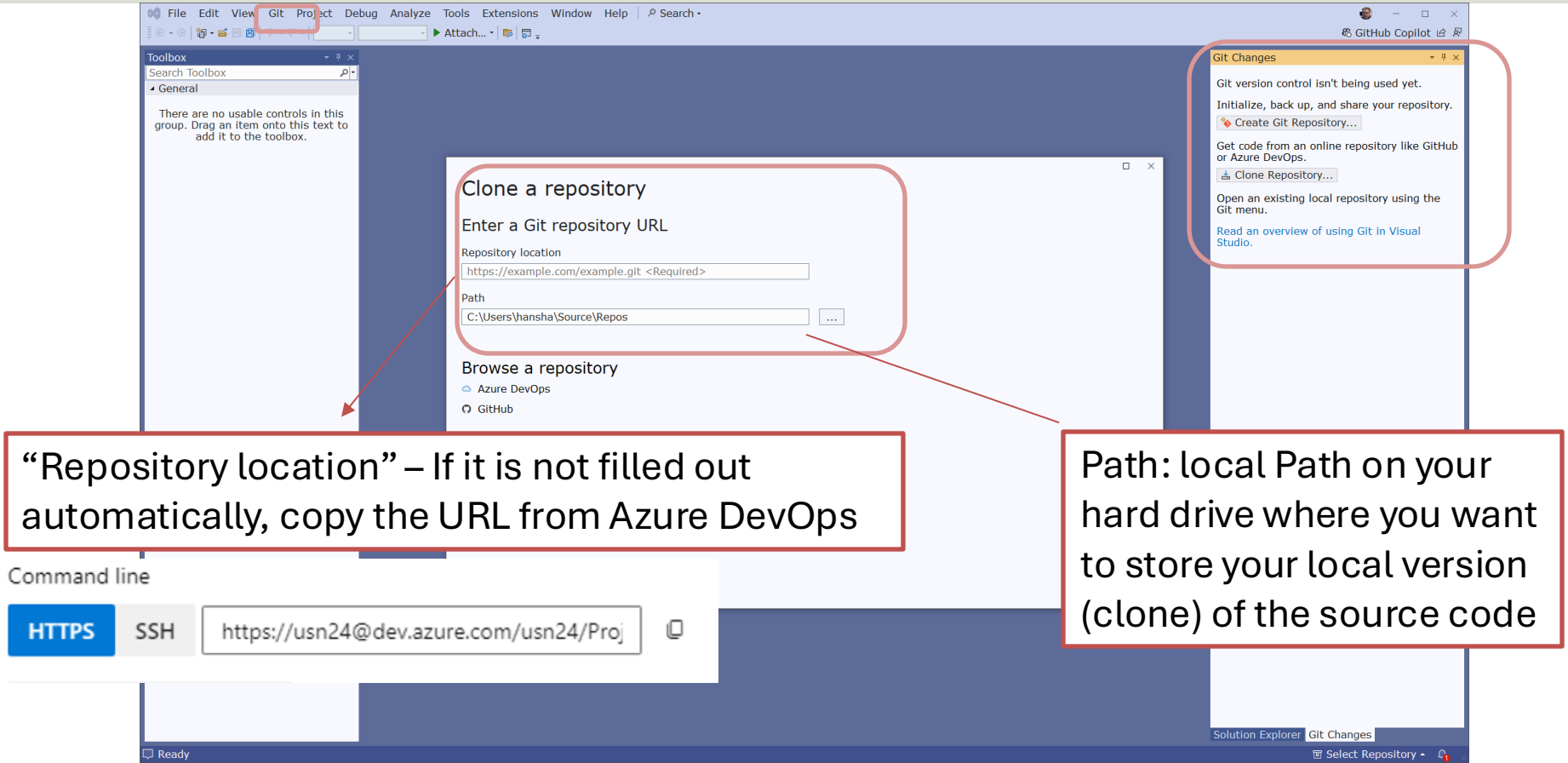
Project URL: https://usn24@dev.azure.com/usn24/ProjectHPH/_git/ProjectHPH







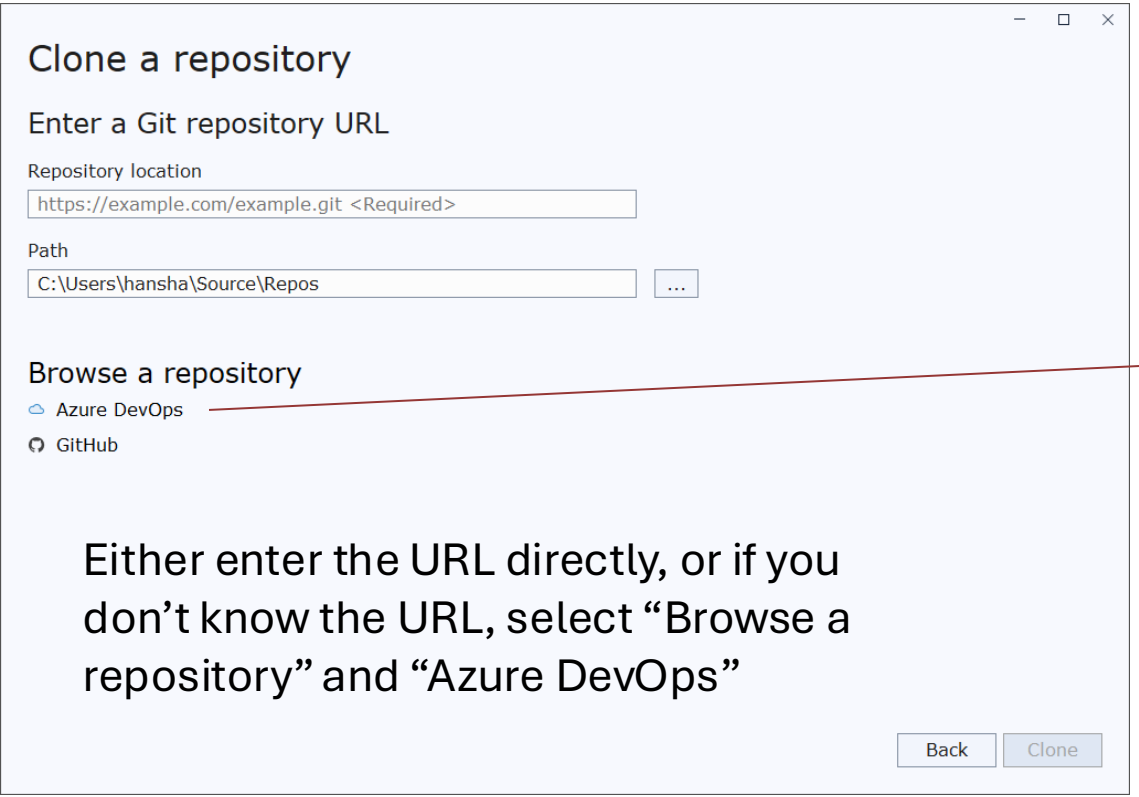
Visual Studio – Clone a repository

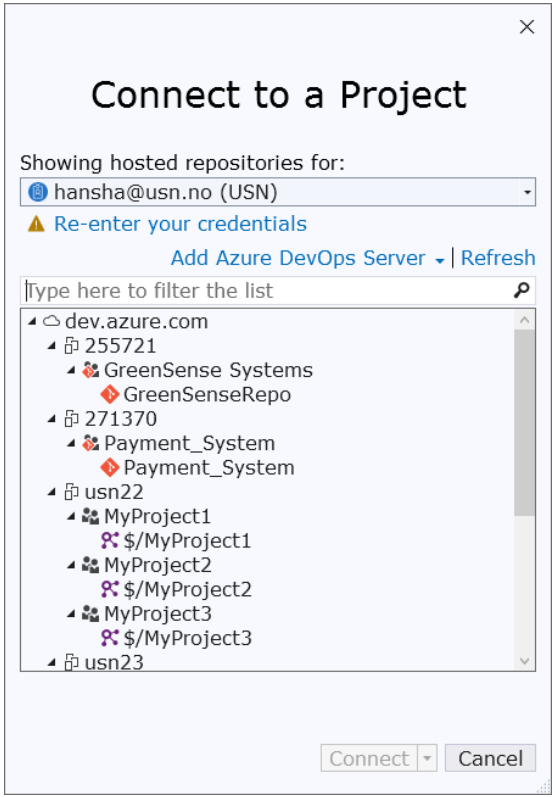


"Repository location" – If it is not filled out automatically, copy the URL from Azure DevOps

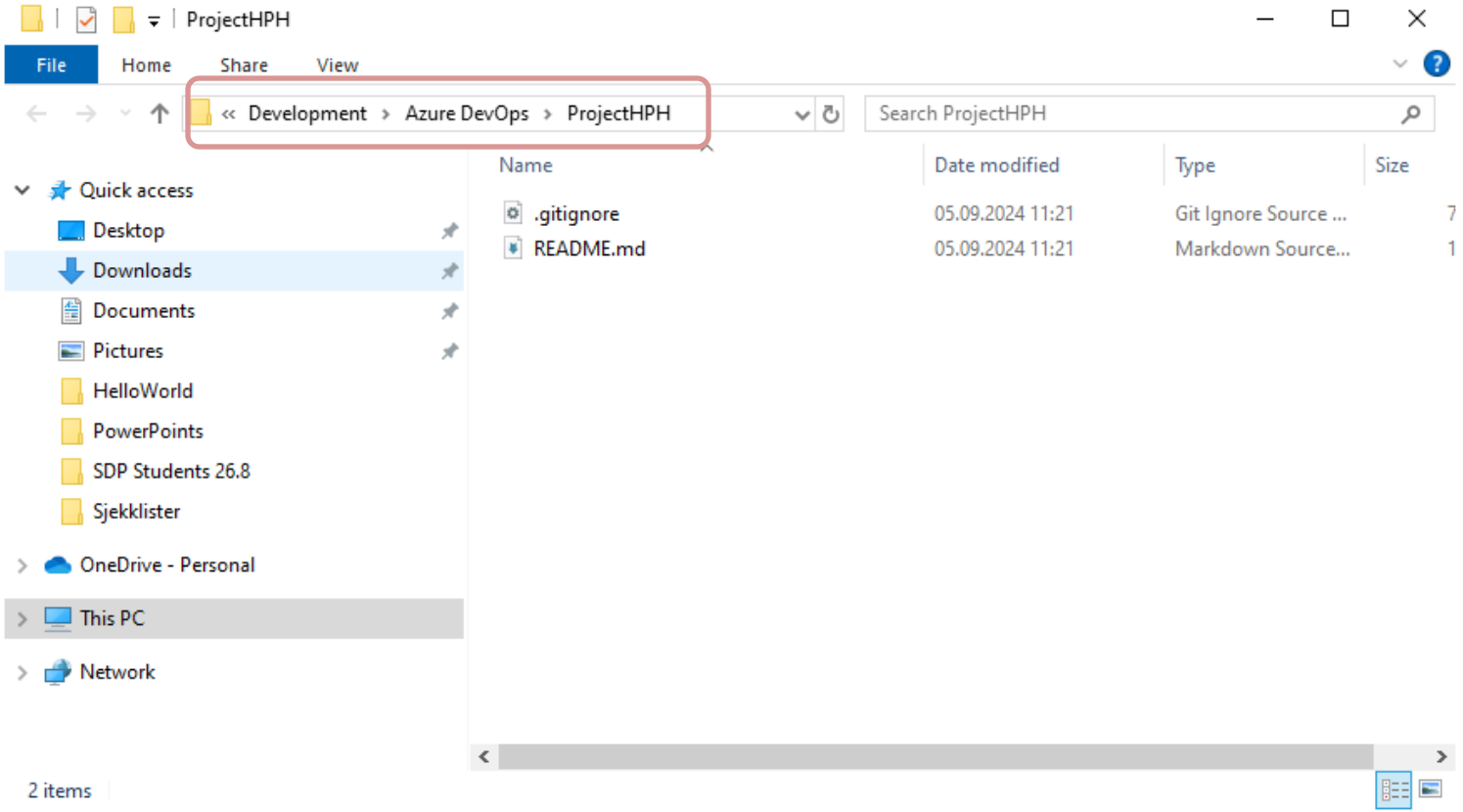
Path: local Path on your hard drive where you want to store your local version (clone) of the source code

Clone a Repository

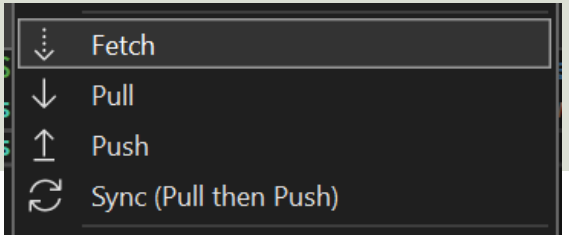




Local Clone/Development folder



Git Terms

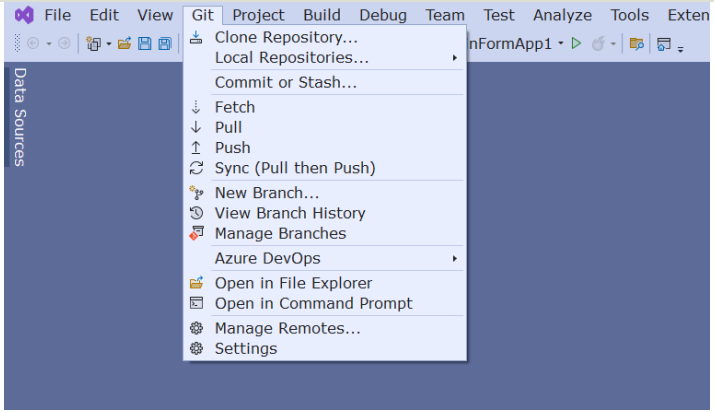


Git maintains a local copy of the entire repository, so you need now and then to sync your local activities with the server.

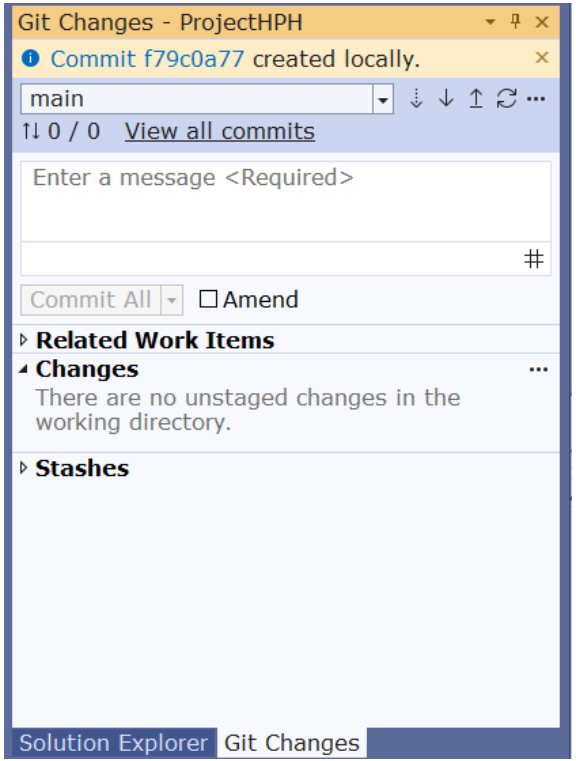
- **Fetch** - It's important to fetch and pull before you push.
  - Fetching checks if there are any remote commits that you should incorporate into your local changes. If you see any, pull first to prevent any upstream merge conflicts.
- **Pull** - Always pull before you push. When you pull first, you can prevent upstream merge conflicts.
- **Push** - When you create **commits** you save local snapshots of your code. You then need to use **Push** to push the commits to the server.
- **Sync** - Use this operation to both Pull, then Push, sequentially.

Git Tools in Visual Studio

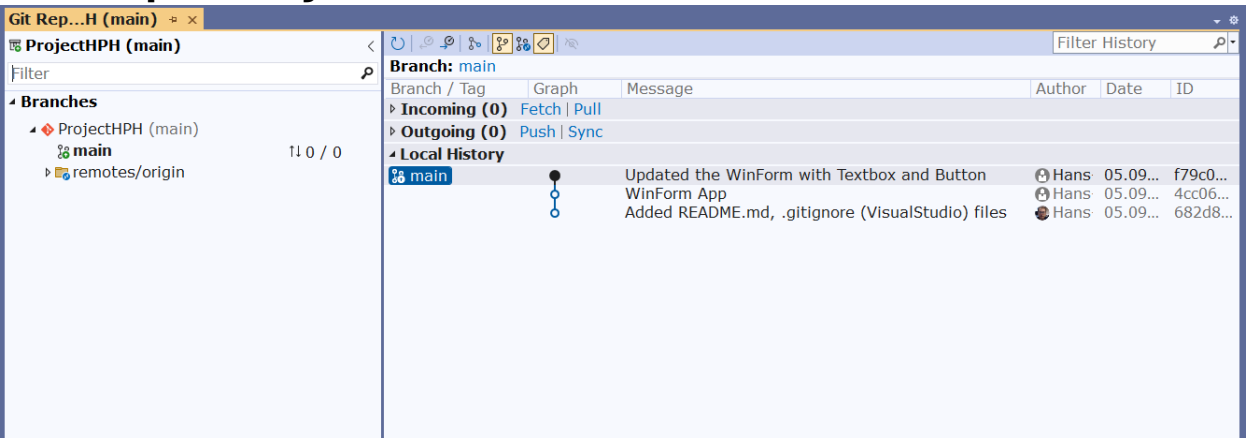
Git menu



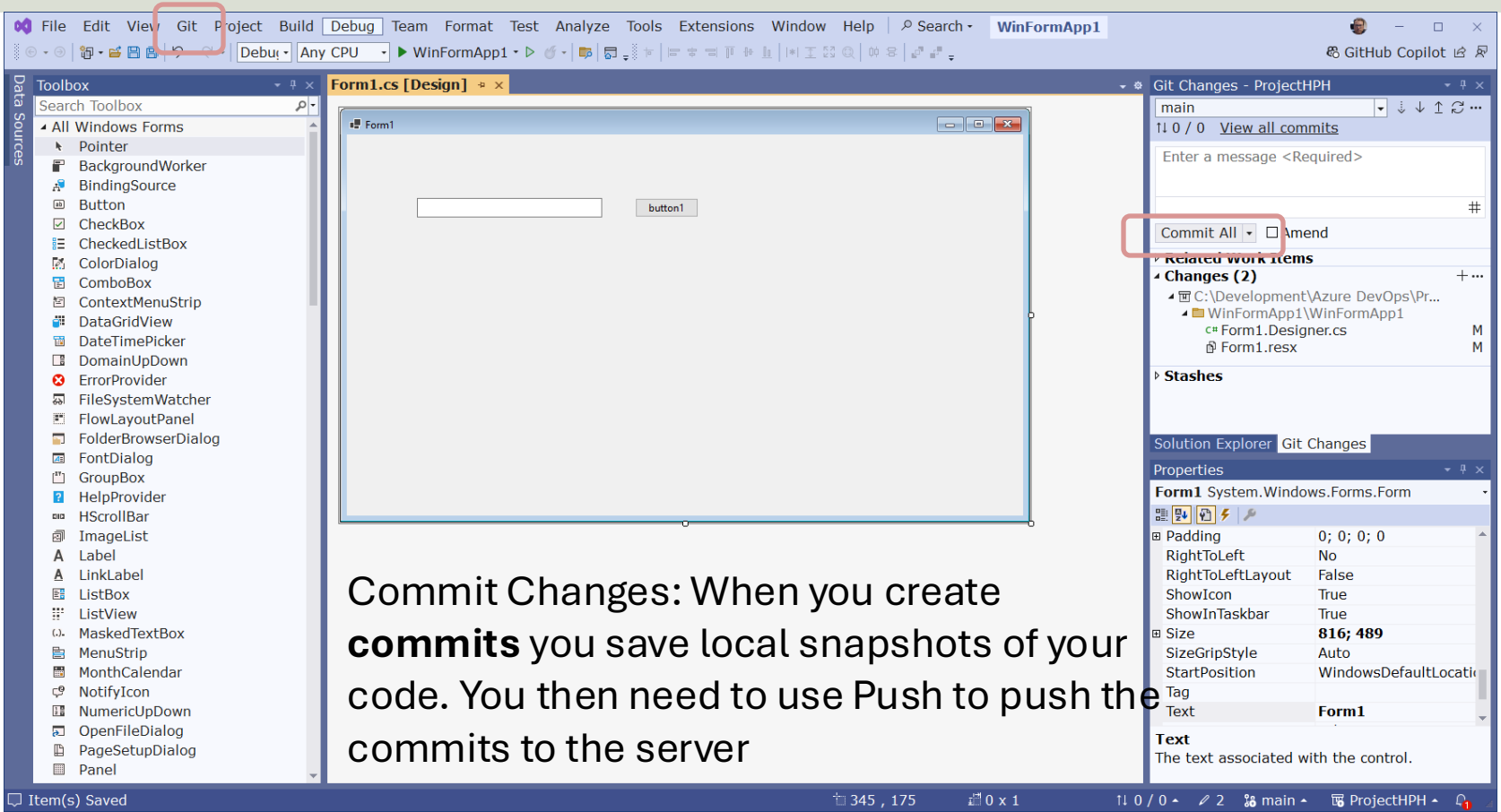
Git Changes window



Git Repository window

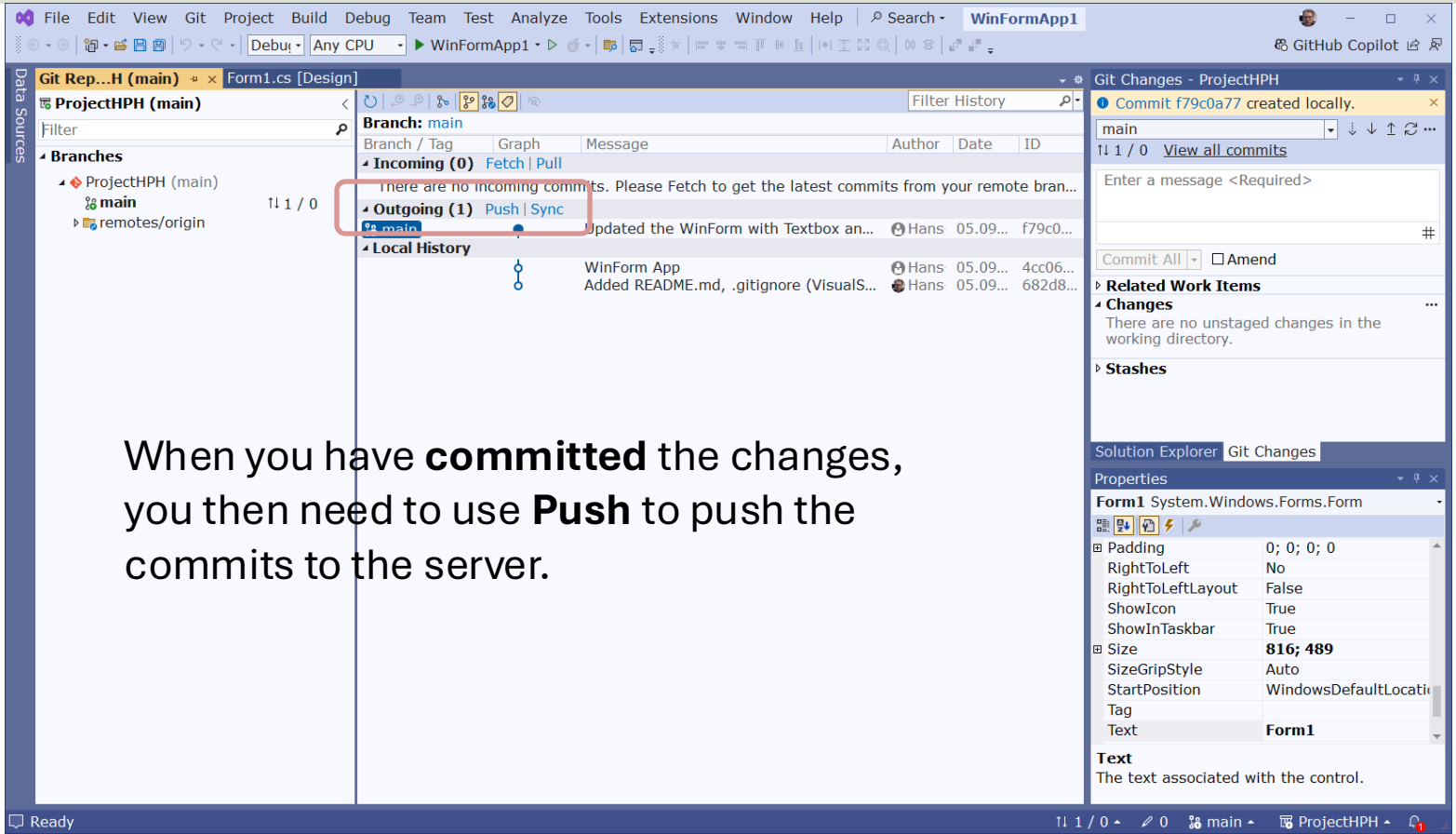


Commit Changes



Commit Changes: When you create **commits** you save local snapshots of your code. You then need to use Push to push the commits to the server

Push Changes

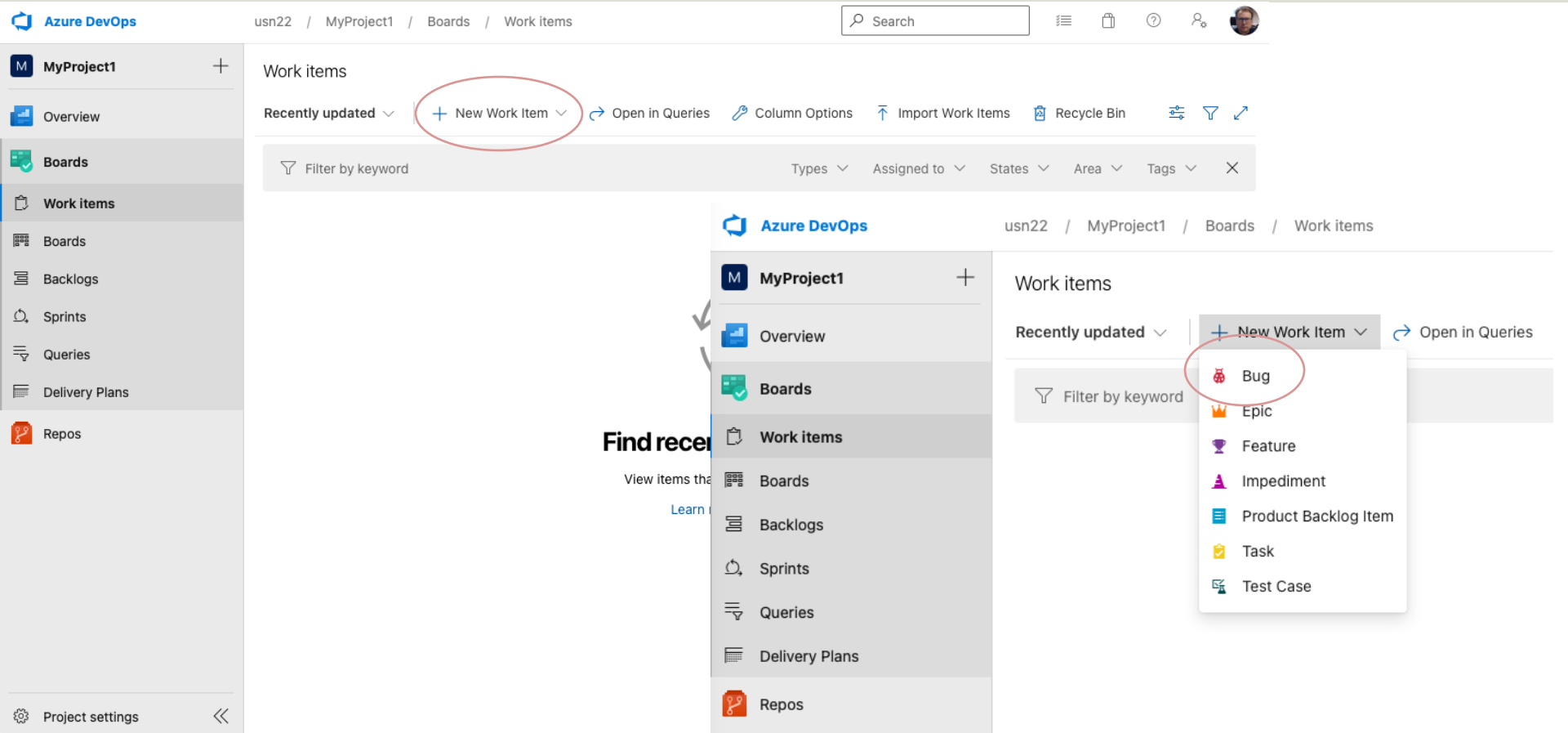


When you have **committed** the changes, you then need to use **Push** to push the commits to the server.

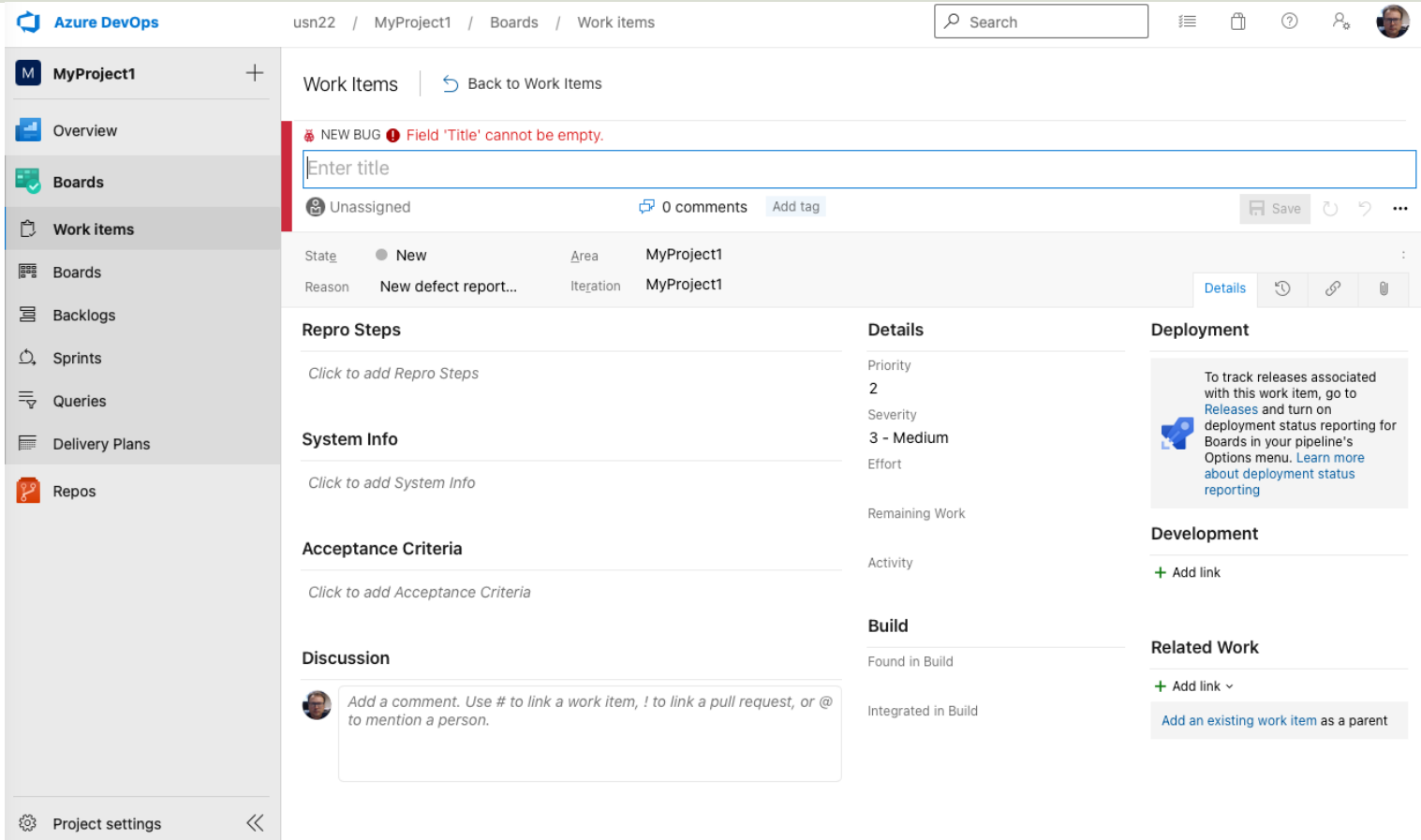
Bug Tracking with Azure DevOps

Hans-Petter Halvorsen

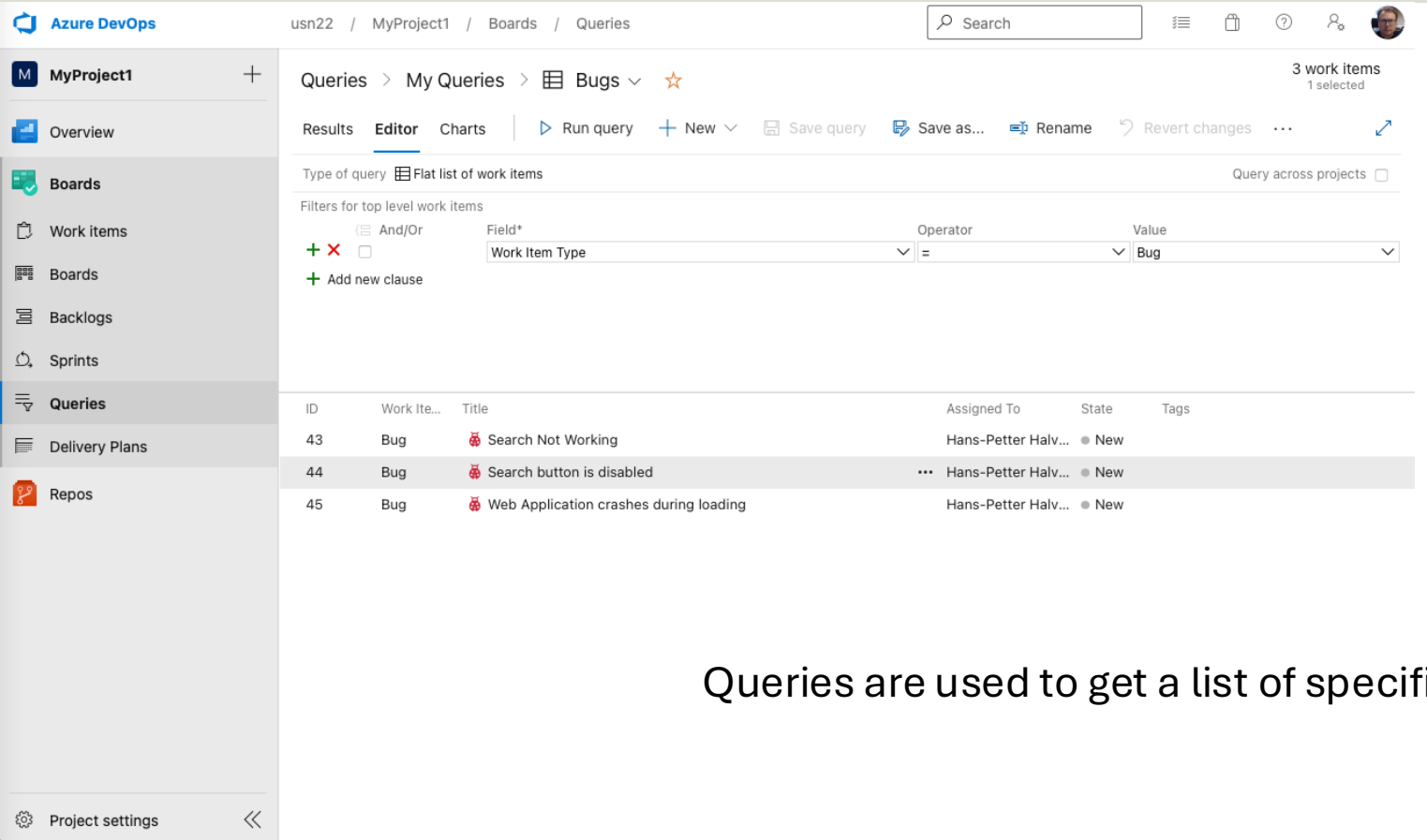
Work Items


Azure DevOps
usn22 / MyProject1 / Boards / Work items
Search
MyProject1
Overview
Boards
Work items
Boards
Backlogs
Sprints
Queries
Delivery Plans
Repos
Project settings
Work items
Recently updated
New Work Item
Open in Queries
Column Options
Import Work Items
Recycle Bin
Filter by keyword
Types
Assigned to
States
Area
Tags
Bug
Epic
Feature
Impediment
Product Backlog Item
Task
Test Case

Bug Reporting



Queries



Queries are used to get a list of specific Work Items

Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

E-mail: hans.p.halvorsen@usn.no

Web: https://www.halvorsen.blog