



## Outline



### Why & what

- Traditional IT
- Continuous Delivery
- DevOps

### How

- GitHub
  - Repos
  - Issues, Projects and Pages
  - Actions
  - Package management
  - Supply chain tools
  - Code Spaces

### Summary



# Why?

Paradigm shift from the waterfall way of work towards an agile way of work

We need a toolset that can help us provide insights in a holistic way on all the steps involved building and running software.

Secure, traceable, reliable, easy to use and no maintenance on the tools yourself!



## Traditional vs. Modern Software Delivery

Traditional	Modern
Waterfall approach	Agile, often scrum approach
Different teams or organizational units for requirements, development, test and operations	Multidisciplinary teams where all disciplines work together on small pieces to deliver
Clear separation between business and IT (demand/supply)	Business, development and operations in one team
Release software 2 or 3 times a year	Release multiple times a day
Budget/cost driven	Value stream driven



“

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software

— 1<sup>st</sup> principle behind agile manifesto

”



## Continuous Delivery

Continuous delivery is all about creating a repeatable and reliable process for delivering software in order to deliver high value software to our customers fast!



“

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

—Donovan Brown

”




Awesome, but how do we do this?


In a secure and compliant way?




## Meet the GitHub Toolset




Project/Product Management




Automation



Package Management

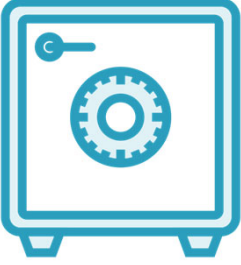


Software Supply Chain

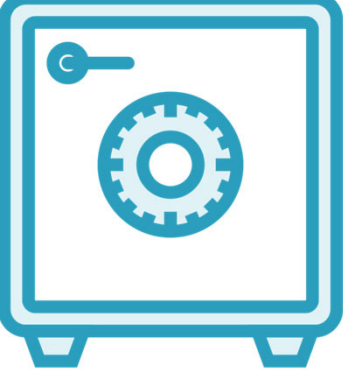


Work From Anywhere

Source Control



## Source Control



Git repo

Public / Private

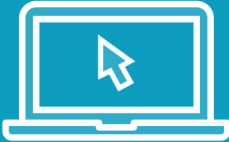
Pull requests

Branch policies


Triggers for automation

T04 - Azure and GitHub the Big Picture - Marcel de Vries

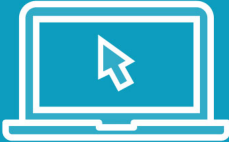
Dem o




Repos

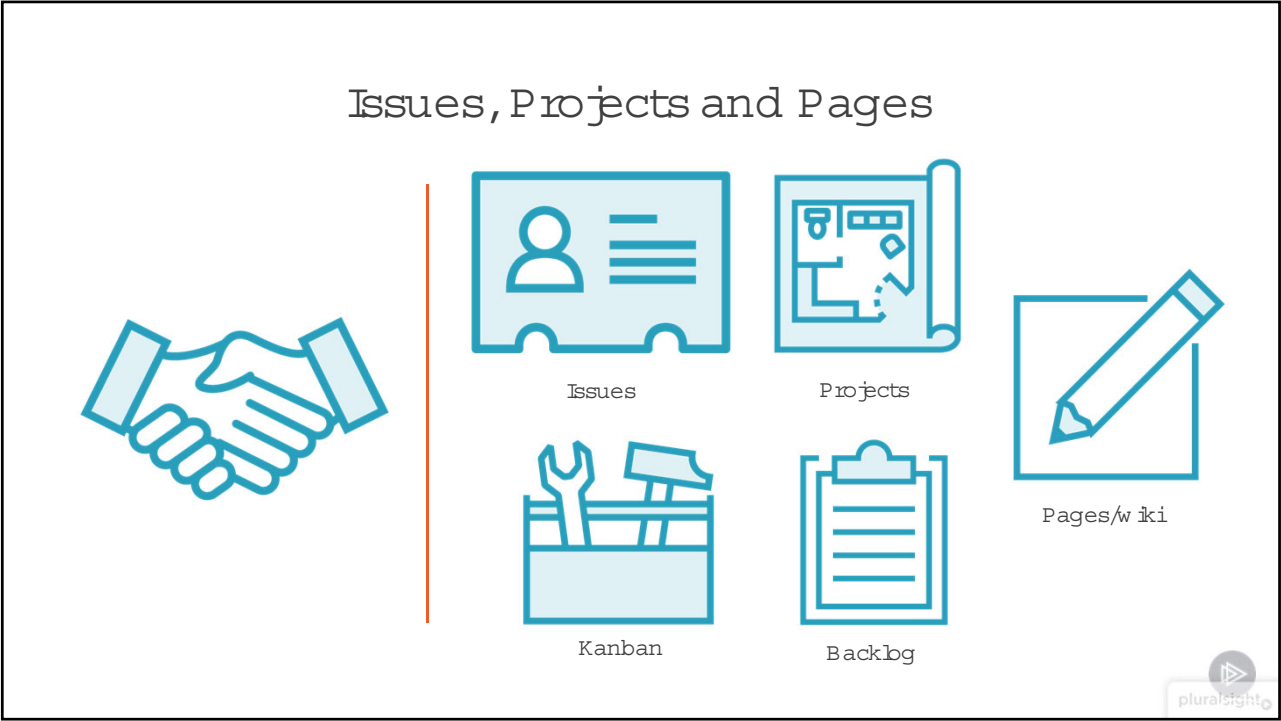


Dem o



PullRequests

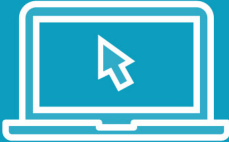





Demo

GitHub Issues

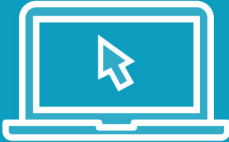
Dem o




G i t H u b P r o j e c t s




Dem o



G i t H u b P a g e s








# Automation


- GitHub Actions
- Continuous integration
- Deployment automation
- Continuous delivery
- Traceability and compliance
- Hooks for other products



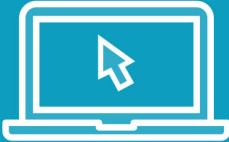
# Demo




# PR Verification




Dem o




Deploy to Azure





### GitHub Packages

- Package registry
- A standard package manager
  - NPM (NodeJS)
  - NuGet (NET)
  - RubyGems (Ruby)
  - Maven and Gradle (Java)
- ContainerRegistry
- Unified Identity and permissions

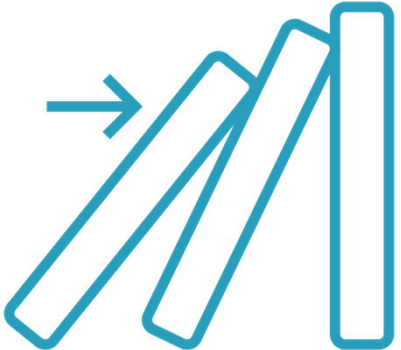


Dem o



G i t H u b Packages






Softw are Supply Chain


Your softw are is build on other softw are

- Has dependencies
- Has know n vulnerabilities

Keep it secure


- Keep it up to date
- Scan for know n vulnerabilities

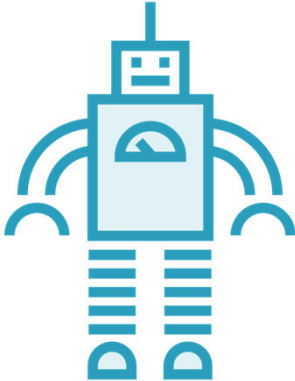




## GitHub Advanced Security


- Part of GitHub Enterprise
- Code Scanning
  - CodeQL
  - SARIF for 3rd party tools
- Secret Scanning
  - Many secrets of known service providers





## Dependabot

- GitHub tooling to keep dependencies up to date
  - Outdated Packages
  - Vulnerable Packages
- Actions in your workflows




Dem o



Code Scanning and Dependabot





# Codespaces


Developm ent environm ent hosted in the cloud

D ifferent VM sizes

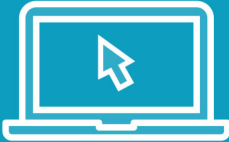
Brow ser or V isual Studio Code

devcontainer.json


Custom container




## Demo



## Codespaces



## Summary



### Why & what

- Traditional IT
- Continuous delivery
- DevOps

### How

- GitHub
  - Repos
  - Issues, Projects and Pages
  - Actions
  - Package management
  - Supply chain tools
  - Code Spaces

