معاونت توسعه و برند

واحد نرم‌افزار

**«کتابچه جامع Azure DevOps»**



فهـرســت

جلسه 1

[9](#_Toc74398273)

[Microsoft Azure DevOps What is 9](#_Toc74398274)

[How does Azure DevOps interact with other tools? 9](#_Toc74398275)

[Azure Boards 9](#_Toc74398276)

[10](#_Toc74398277)

[Azure Repos 10](#_Toc74398278)

[Azure Pipelines 10](#_Toc74398279)

[Azure Test Plans 10](#_Toc74398280)

[Azure Artifacts 11](#_Toc74398281)

[The Punchline 11](#_Toc74398282)

[Roles 12](#_Toc74398283)

[Teams 12](#_Toc74398284)

[Check the tool itself 13](#_Toc74398285)

[Technical Dept 14](#_Toc74398286)

[Rules 14](#_Toc74398287)

[Check Team details 15](#_Toc74398288)

[Area 15](#_Toc74398289)

[Dashboard 15](#_Toc74398290)

[Wiki 16](#_Toc74398291)

[Global notification 18](#_Toc74398292)

[Extension 18](#_Toc74398293)

[Analytics 18](#_Toc74398294)

[Agent tools 18](#_Toc74398295)

[Retention 19](#_Toc74398296)

[Security 19](#_Toc74398297)

[نتیجه 20](#_Toc74398298)

[مقدمه 22](#_Toc74398299)

[Work item‌های مورد نیاز 22](#_Toc74398300)

[فیلدهای جدید مورد نیاز به شرح زیر هستند : 23](#_Toc74398301)

[نوع Product 23](#_Toc74398302)

[لایه Backlog Items 24](#_Toc74398303)

[چه زمانی Type of Systemرا اضافه نماییم؟ 25](#_Toc74398304)

[نوع Support 25](#_Toc74398305)

[نوع Task 25](#_Toc74398306)

[نوع Task Est 26](#_Toc74398307)

[توضیح Acceptance Criteria 26](#_Toc74398308)

[دیدگاه اسکرام 27](#_Toc74398309)

[در لایه Task 27](#_Toc74398310)

[در لایه Bug 27](#_Toc74398311)

[مقدمه 30](#_Toc74398312)

[مدیریت محصول در Azure DevOps 30](#_Toc74398313)

[Board 30](#_Toc74398314)

[زیرمجموعه‌های Board 30](#_Toc74398315)

[متدولوژی KanBan 31](#_Toc74398316)

[Type Of Queries 33](#_Toc74398317)

[بررسی یک Sample برای Board 33](#_Toc74398318)

[Extention Delivery Plans چه امکانتی را فراهم می‌کند؟ 33](#_Toc74398319)

[مفهوم User Story 35](#_Toc74398320)

[Mapping 37](#_Toc74398321)

[نکاتی در خصوص KanBan 37](#_Toc74398322)

[**«Microsoft Azure Devops (session 5)»** 38](#_Toc74398323)

[39](#_Toc74398324)

[مقدمه 39](#_Toc74398325)

[بررسی ساختار پروژه ها در BackLog 39](#_Toc74398326)

[اگر Area های هر Product مشخص نشود، در آینده در گرفتن کوئری ها دچار مشکل خواهید شد، زیرا نمی توان مشخص کرد که هر قسمت مربوط به کدام محصول می باشد. 39](#_Toc74398327)

[دیاگرام های Velocity و Cumulative Flow Diagram 40](#_Toc74398328)

[40](#_Toc74398329)

[بررسی Sprint ها 40](#_Toc74398330)

[41](#_Toc74398331)

[بررسی Queryها 41](#_Toc74398332)

[نحوه نوشتن شرط در Query 41](#_Toc74398333)

[42](#_Toc74398334)

[بررسی Dashboard 42](#_Toc74398335)

[بررسی Widget ها 43](#_Toc74398336)

[نکات 43](#_Toc74398337)

[46](#_Toc74398338)

[مقدمه 46](#_Toc74398339)

[بررسی WIKI 46](#_Toc74398340)

[بررسی مفهوم Query Result 49](#_Toc74398341)

[معمولا می توان Wiki را در سه لایه ایجاد نمود 49](#_Toc74398342)

[مبحث REPO 50](#_Toc74398343)

[فرآیند انتقال Git Repository به Repo در Azure 51](#_Toc74398344)

[53](#_Toc74398345)

[مقدمه 53](#_Toc74398346)

[Branches یا انشعاب ها 53](#_Toc74398347)

[برنچینگ مدل های استاندارد 54](#_Toc74398348)

[شیوه و اصول ایجاد انشعاب ها: 57](#_Toc74398349)

[59](#_Toc74398350)

[Branch policy 59](#_Toc74398351)

[Pull Request یا درخواست ادغام 61](#_Toc74398352)

[ارتباط بین Remote Ripository و Local Repository 63](#_Toc74398353)

[Tags 63](#_Toc74398354)

[نکات استفاده از ویژوال استودیو برای مدیریت ریپازیتوری 64](#_Toc74398355)

[امنیت در Repositoryها 65](#_Toc74398356)

[یکپارچه سازی خط مشی های مرتبط با تمامی ریپازیتوری ها بر اساس یک Branching Model خاص 66](#_Toc74398357)

[Pipeline Azure 68](#_Toc74398358)

[CI (Continuous Integration) 68](#_Toc74398359)

[CD (Continuous Delivery (Deployment)) 68](#_Toc74398360)

[کانسپت های کلیدی 69](#_Toc74398361)

[Trigger 69](#_Toc74398362)

[Stage 70](#_Toc74398363)

[Approval 71](#_Toc74398364)

[Job 71](#_Toc74398365)

[Agent 71](#_Toc74398366)

[Pipeline 71](#_Toc74398367)

[Environment 72](#_Toc74398368)

[Deployment group 72](#_Toc74398369)

[Step 72](#_Toc74398370)

[Script 73](#_Toc74398371)

[Task 73](#_Toc74398372)

[Run 73](#_Toc74398373)

[Artifact 73](#_Toc74398374)

[استفاده از Classic interface 74](#_Toc74398375)

[استفاده از YAML syntax 74](#_Toc74398376)

[دانلود و نصب Agent جهت اجرای Pipeline 75](#_Toc74398377)

[آشنایی با بخش Task groups 76](#_Toc74398378)

[آشنایی با بخش Library 76](#_Toc74398379)

[معرفی بخش Pipeline 78](#_Toc74398380)

[نکاتی درباره Get resource 78](#_Toc74398381)

[Jobs 80](#_Toc74398382)

[Tasks 81](#_Toc74398383)

[Variables 82](#_Toc74398384)

[History 83](#_Toc74398385)

[83](#_Toc74398386)

[سلسله مراتب پیاده سازی یک فایل YAML 83](#_Toc74398387)

[Status Badge 84](#_Toc74398388)

[معرفی بخش Environment 86](#_Toc74398389)

[Deployment job 86](#_Toc74398390)

[تبدیل Classic pipeline به YAML 87](#_Toc74398391)

[استفاده از Template به جای Task group در YAML 87](#_Toc74398392)

[تنظیمات Agent Pool 88](#_Toc74398393)

[Maintenance job 89](#_Toc74398394)

[Maintenance history 90](#_Toc74398395)

[Secure fileها در بخش library 90](#_Toc74398396)

[**«Microsoft Azure Devops (session 13)»** 91](#_Toc74398397)

[شرح CICD در آژور 92](#_Toc74398398)

[وضعیت های Build در تب Runs 94](#_Toc74398399)

[**«Microsoft Azure Devops (session 14)»** 95](#_Toc74398400)

[گزارشات در Pipelineها 96](#_Toc74398401)

[Retention Policy 97](#_Toc74398402)

**«Microsoft Azure DevOps (Session 1)»**

# **Microsoft** **Azure** **DevOps** **What is**

این ابزار با استفاده از ماژول های متنوع خود چرخه عمر توسعه نرم افزار را در بر میگیرد تا به تیم ها کمک کند محصول خود را سریعتر و با کیفیت بالاتر مدیریت و توسعه نمایند. ماژول های Azure DevOps شامل موارد زیر است:

1. Azure Board (مدیریت محصول و Task‌ها و تست‌های Manual)
2. Azure Repos (نگهداری کدها و مدیریت ورژن کنترلینگ نرم افزار - با Azure Boards، ارتباط مستقیمی دارد)
3. Azure Pipelines (پیاده سازی چرخه CI/CD با هدف صرفه‌جویی در زمان برای Update کردن IoS یا Web Server و...)
4. Azure Test Plans (پیاده سازی و مدیریت تست‌هایی مثل Manual test، Automated test و...)
5. Azure Artifacts (نگهداری و کش نمودن Package‌ها)

این مجموعه توانایی های غنی از جمله پشتیبانی از Scrum و Kanban را بصورت بومی داشته و داشبورد قابل تنظیم و گزارش گیری یکپارچه ای را فراهم می کند.

# **How does Azure DevOps interact with other tools?**

این ابزار تعامل و سازگاری زیادی با سایر نرم افزارهای مرتبط داشته و میتواند با زبان‌های برنامه‌نویسی مختلف از قبیل #C، VB.NET و... تعامل برقرار نماید.



Azure Boards

تیم های مختلف با استفاده از این ماژول میتوانند پروژه های نرم افزاری خود را مدیریت و اقدامات تعیین شده را ردیابی نمایند.   
برای مثال میتوان چگونگی تعامل تیم‌های مختلف (UI، UX، Front و ...) را بر روی کارهای مختلف سازمان پیاده سازی نمود.

# 

Azure Repos

Azure Repos یک سرویس ورژن کنترلینگ بوده که می توانید برای مدیریت کد خود استفاده کنید. برای صرفه جویی در کارها و ایجاد هماهنگی در پروژه توسعه نرم افزار، از این سرویس استفاده کنید. حتی اگر فقط یک توسعه دهنده هستید ، کنترل نسخه به شما کمک می کند هنگام رفع اشکالات و ایجاد ویژگی های جدید ، با نظم و کیفیت بهتری اقدام نمائید.



Azure Pipelines

جهت پیاده سازی سیستم های CI/CD مورد استفاده قرار میگیرد. به بیان ساده تر در برخی پروژه ها می‌توانیم با استفاده از کدهای Power Shell، Bash و ... اجرای بخشی از اقدامات توسعه ای سازمان یا کل آن را اتوماتیک نماییم.

# **Azure Test** **Plans**

این سرویس مرتبط با Capture کردن تست‌ها می‌باشد. همچنین از طریق آن می‌توانید گزارشی از Bug‌هایی که داشتید و این که این Bug‌ها در چه Level‌ی بودند و ... تهیه نمایید، که در برنامه ریزی برای محصول جدید مؤثر می‌باشد.

****Azure Artifacts

در اصل نگهدارنده Package های مورد استفاده و خروجی های مرتبط با نرم افزارهای توسعه داده شده می باشد.



The Punchline

در این قسمت اشاره ای به داستان Microsoft می‌نماییم. در اصل هدف اصلی داشتن تیم‌هایی است که به خود به خود بتوانند Feature‌هایی که در حال توسعه آن‌ها هستند را مدیریت نمایند. هم چنین Sprint‌های آن‌ها 3 هفته‌ای بوده و برنامه‌ریزی آن‌ها به صورت دوطرفه می‌باشد.

در اصل هدفی که Microsoft در نظر گرفت این بود که به جای این که مثلا 6 ماه برای تولید یک محصول زمان بگذاریم، می‌توانیم در مدت 3 ماه ارائه‌ای از نسخه اولیه آن محصول داشته باشم، که همان مفهومScrum و Agile می‌باشد.

مسئله ای که وجود داشت دوساله بودن Planning‌های آنان بود، یعنی دو ماه تلاش می‌کردند که Planning انجام دهند و مابقی زمان صرف Milestone‌ها می‌شد.

یعنی 8 تا 12 هفته را صرف کدنویسی می‌کردند و مابقی زمان را صرف تست و Stable کردن سیستم می‌نمودند و زمانی که سیستم Stable می‌شد دوباره کدنویسی می‌کردند و و مابقی زمان صرف تست و Stable کردن می‌شد. پس منطقی نیست که ما بدون در نظر گرفتن کیفیت کدنویسی نماییم و مدت زمان زیادی را صرف از رفع Bug‌ها و بهم ریختگی‌ها نماییم.

مسئله دیگر Feed یا بازخوردی است که مشتری یا تیم‌های دیگر پس از برنامه‌ریزی کارها می‌دهند و ما زمان کافی برای اعمال آنها نداریم، و درنتیجه محصولی را ارائه می‌دهیم که مشتری از آن راضی نمی‌باشد.

به دلایل زیر Microsoft به این نتیجه رسید که روش قبل برای توسعه دادن درست نمی‌باشد:

1. زمان کافی برای رفع نیاز مشتری را نداشتند.
2. کارمندها ناراضی بودند.
3. در نظر داشتند که به سمت Cloud بروند.

درواقع وجود این Sprint‌های 3 هفته‌ای باعث افزایش روند Develop می‌شود.



Roles

در حقیقت ساختار تیم‌ها بسیار مهم است. در ایران معمولا 3 نوع تیم وجود دارد:

* تیم test
* تیم develop
* تیم مدیریت

Microsoft در ابتدا از این ساختار پیروی می‌کرد ولی مشاهده کرد نتیجه‌ی مطلوبی را ایجاد نمی‌کند، پس تصمیم گرفتند تا ساختار تیم‌ها را عوض نمایند. در اصل به این نتیجه رسیدند که جدا بودن تیم تست درست نمی‌باشد، پس تیم مهندسی را تشکیل دادند که هر عضوی از این تیم وظیفه توسعه و تست را به طور همزمان بر‌عهده داشت. همچنین برای رفع نیاز مشتری تیم دیگری وجود دارد ((program managemen که با مشتری‌ها تعامل و گفتگو می‌نمایند تا نیازشان را رفع کنند.



Teams

# 

بهینه‌ترین روش وجود تیم‌های Scrum است به نحوی که خود آن تیم بتواند تمام کارهای خودش را انجام دهد. این بحث در چیدمانAzure

DevOps تاثیر می‌گذارد.

**Check the tool itself**

* DefaultCollection
* Azure سعی می‌کند برای مدیریت سازمان‌ها آنها را در 3 لایه بشکند که اولین لایه Collection است (بالاترین سطح برای جدا کردن کارها از یکدیگر).
* پایین تر از لایه Collection لایه دیگری به Project (Team Project) وجود دارد. در نظر داشته باشید که برای مثلا یک تیم 10 نفره نباید یک Team Project ایجاد نماییم.

Collection => board => backlogs => مشاهده تیم‌های اسکرام

* برای حفظ ابزار از بهم‌ریختگی تیم‌های Scrum و Agile را به وجود آورید.
* نباید به ازای هر پروژه‌ای در Collection یک Team Project ایجاد نماییم و همچنین باید ساختار تیم‌های اسکرام را حفظ نماییم. چنان چه تعداد Team Project‌ها زیاد شود، نگهداری آنها مشکل می‌شود. پس رویکرد داشتن یکTeam Project می‌باشد.

Collection Setting => بالاترین سطح تنظیماتی است که می‌توانیم داشته باشیم

Basic (default)

Agilee

ابزار Azure DevOps 4 فرآیند برای استفاده از ابزار

Process

Scrum

در اختیار ما قرار می‌دهد.

CMMI

* اگر روی Scrum کلیک نماییم Detail زیر را مشاهده خواهیم نمود:
* Bug
* Epic
* Feature
* Impediment
* Product backlog item
* Task
* Level‌ها بسیار حائز اهمیت هستند.
* Epic (کار بلند مدت)
* Feature (کاری که ممکن است دو یا سه ماه طول بکشد)
* Backlog item (کارهایی که در یک Sprint باید انجام شوند)
* Task ( تنها لایه‌ای که زمان و Time در آن دخیل می‌باشد)
* شما Collection به Collection می‌توانید فرآیندهای مختلفی داشته باشید، ولی نگهداری آن مشکل می‌شود.
* گاهی برای انجام یک پروژه‌ای که در 3 ماه باید به پایان برسد، پس از گذشت دو یا سه هفته متوجه می‌شویم که این پروژه 9 ماه زمان می‌برد، در این موقعیت باید تصمیم بگیریم که پروژه را کنسل نماییم یا ادامه دهیم که با آن Faild Fast (شکست سریع) می‌گویند.



Technical Dept

Technical Dept یا بدهی فنی مجموعه مشکلات موجود در مسیر توسعه نرم افزار است که به پیشرفت ناپایدار ارزش مشتری دامن میزند.  
یک Product Owner برای چیدن backlog محصول باید موارد زیر را بداند:

* چه میزان Technical Dept باید در Sprint مورد نظر برطرف شود.
* چه میزان Feature باید Deliver شود.
* چه میزان Bug باید برطرف شود.
* و ...

Technical Dept توانایی این را دارد که یک تیم IT را زمین‌گیر نماید، یک تیم IT را حذف نماید یا محصولی را با شکست مواجه کند.



Rules

به وسیله آنها می‌توان یکسری مسائل را محدود نمود. از آن جایی که به مرور زمان تعداد آنها زیاد می‌شود، بهترین الگویی که می‌توان استفاده نمود شروع اسم هر قانون با State می باشد، زیرا هر Rule بر روی یک State قرار دارد. به مثال زیر توجه نمایید:

اگر State work item از نوع New بود فیلد assign to آن اجباری شود.

New-assign to-Required



Check Team details

مفهوم Team در Project Setting که سومین لایه مدیریتی در Azure DevOps است، تیم های Scrum می باشد. (تیم‌های که حداکثر تعداد آنها بین هفت تا هشت نفر می‌باشد و در زمینه‌های مختلفی تخصص دارند و از پس کارهایی که به آنها محول می‌نمایید بر می‌آیند.)

* Team Project‌ها از یکدیگر جدا هستند و کسی کارهای Team Project دیگری نمی‌بیند، همچنین Repository‌های یکدیگر را نمیتواند ببیند و قادر به مشاهده Pipeline یکدیگر نیستند. بنابراین ساختن چیدمان براساس نیاز سازمان که چه زمانی افراد بتوانند کارهای یکدیگر را مشاهده نمایند و چه زمانی قادر به مشاهده کارهای یکدیگر نباشند، بسیار حائز اهمیت است.



Area

به ما کمک میکند تا Work item ها را بر اساس تیم ها یا محصولات دسته بندی کنیم. بطور کلی از این امکان جهت دسته بندی و تعیین محصولات سازمان استفاده می شود. (به عبارتی دیگر به وسیله Area محصولات را می شکنیم)



Dashboard

فرض کنید از شما تقاضا شده است تا پروژه‌ها را چک نموده و گزارشی تهیه نمائید، برای انجام این کار فقط کافی است که پروژه ما دارای یک Dashboard باشد تا در آن موارد زیر را نشان دهیم:

* در Sprint جاری و گذشته چه کارهای انجام شده است.
* در Sprint جاری و گذشته چه کارهایی Deliver شده است.
* در Sprint جاری و گذشته چه تست‌هایی Faild شده است.
* و ...

با استفاده از پیاده سازی Query ها و فراخوانی آنها در داشبورد تقریباً هر نوع گزارشی قابل پیاده سازی است.   
در ادامه به Queryها خواهیم پرداخت.



Wiki

می‌توانید برای Share اطلاعات مانند Data‌ها، File‌ها، Voice recorder‌ها و ... به جای استفاده از UNC Share از Wiki استفاده نمایید.

**»Microsoft Azure DevOps (Session2) «**

در این فصل قصد داریم به دسترسی‌های تیم‌ها در دواپس بپردازیم.

در ابتدا به این نکته توجه داشته باشید که برای استفاده از تمامی بخش های Azure DevOps در قسمت Collection> Collection Setting> Access level> Default را روی گزینه Vs Enterprise قرار دهید.



Global notification

در این قسمت تعیین می‌گردد که به ازای اتفاقاتی که در سطوح مختلف رخ می‌دهد، ایمیل Notification دریافت شود.



Extension

میتوان Extension‌های مختلفی را روی Azure DevOps فعال نمود. باید توجه داشته باشید که از Extension‌های خود Microsoft استفاده نمایید سایر Extension‌ها ممکن است باعث کندی یا لاگین کردن‌های متعدد شوند که روند کار با Azure DevOps را مختل کند.

****Analytics

با فعال شدن آنالیتیکس، میتوان بدون استفاده از SSRS و SSIS انواع گزارشات رو از Azure DevOps دریافت کرد.



Agent tools

به منظور اجرای فعالیت های سیستمی که توسط تیم ها برنامه ریزی و زمانبندی شده است از Agentها استفاده می‌شود. به طور مثال برای Run کردن تست‌ها یا بطور کل اجرای فرآیندهای CI/CD



Retention

حجم ماندگاری دیتابیس Azure DevOps با استفاده از این گزینه تعیین می‌شود. با استفاده از این گزینه می‌توان دیتاهایی که مورد نیاز نیستند و قدیمی هستند را حذف نمود.



Security

در ارئه دسترسی‌ها در Azure DevOps نمیتوان به هر یک از کاربران دسترسی دهیم به دلیل اینکه با کلیک روی نام کاربر انتظار داریم مشخص شود که کاربر چه دسترسی‌هایی دارد بلکه تنها می‌توان مشاهده نمود که در چه تیم‌ها و گروه های امنیتی عضو است بنابراین مدیریت دسترسی‌ها توسط شخص غیر ممکن است و می‌بایست با ایجاد گروه‌های امنیتی و انتساب دادن هر کاربر به آن نقش دسترسی‌ها را کنترل کرد.

ایجاد گروه‌های امنیتی در لایه Collection به عنوان بالاترین لایه، دسترسی های کلانی را ارائه می‌دهد مانند حذف کردن Team Project Collection یاTeam Project

برای ایجاد گروه امنیتی در لایه Collection در قسمت Collection Setting در بخش Security روی گزینه New Goup کلیک نمایید و مجوز‌های مورد نیاز با به این گروه امنیتی بدهید.

اما برای مدیریت دسترسی ها در سطوح پایین تر می‌بایست گروه های امنیتی را در لایه project‌ها تعریف کرد. در قسمت project setting در بخش security روی گزینه New Group کلیک نمایید و مجوز‌های مورد نیاز با به این گروه امنتی بدهید. سپس اعضای مورد نیاز را در تب Members به این گروه امنیتی اضافه کرده و در نهایت گروه امنیتی را به Team مربوطه انتساب دهید.

همانطور که می‌دانید، برای هر پروژه Sprint ها و Area های مختلفی می‌توان تعیین نمود؛ شاید نیاز داشته باشد که یک تیم روی Sprint یا Area‌های خاصی دسترسی داشته باشد و اطلاعات مربوطه را مشاهده نماید در این حالت نیز میتوان روی Sprint‌هاو Area ها به گروه های امنیتی مختلف مجوز های مورد نیاز را اعطا کرد. برای این کار در قسمت Project Configuration در بخش مورد نظر (Sprint یا Area) از سه نقطه کنار نام آنها گزینه Security مورد نظر را انتخاب نموده و تیم مورد نظر را انتخاب نمایید.

به طور کلی به منظور ارائه دسترسی به تیم ها به اطلاعات هر sprint یا هر Area دو مورد را بررسی نمایید:

1. تیم را (از قسمت security) به sprint یا Area مورد نظر اضافه کرده باشیم. (به معنی اینکه این تیم روی محصول در حال کار کردن است)
2. به تیم مجوز لازم جهت مشاهده دیتای sprint یا Area را بدهیم.

نتیجه

* برای ارائه دسترسی‌ها بر حسب نیاز از طریق ایجاد گروه‌های امنیتی در لایه collection یا project اقدام شود.
* افراد به گروه های امنیتی اضافه می‌شوند.
* گروه ای امنیتی به تیم ها انتساب داده می‌شوند.
* با استفاده از Area‌ها محصولات شکسته می‌شوند.
* برای ارائه دسترسی به تیم ها به منظور کار روی محصولات، تیم را به Area مورد نظر اضافه می‌کنیم.
* در صورتی که نیاز باشد تیم، به تمامی محصولات مربوط به Area مورد نظر دسترسی داشته باشد تیک فیلد Include Sub Areas را فعال کنید.

**»Microsoft Azure DevOps (Session3) «**



# 

مقدمه

در این سند به بررسی دو موضوع زیر می‌پردازیم:

* آموزش ساختار و نحوه ایجاد تیم‌ها و تیم پروژه‌ها و کالکشن‌ها و مهاجرت به کالکشن جدید ( ادامه موضوع ساختار ابزار و نحوه ایجاد تیم‌های اسکرام و ...)
* آموزش نحوه پیاده‌سازی و نگهداری از بخش محصولات سازمان و اجایل و دانشنامه فنی سازمان (Wiki).

در ادامه به بررسی Feature‌هایی می‌پردازیم که Azure DevOps در رابطه با ارتباط تیم‌های اسکرام و Product Owner و Product Master با یکدیگر، ارائه می‌دهد.

Work item‌های مورد نیاز

برای این منظور که بتوانیم Data هارا درست نشان دهیم و به درستی گزارش تهیه نماییم، و همچنین به دیاگرام ها و چارت هایی که در آینده با آن ها آشنا می شویم کمک نماییم تا به درستی Data‌ها را به ما نشان دهند، بهتر است آن‌ها را تفکیک نماییم:

* **Analysis Design:** برای کارهایی استفاده می‌شود که می‌خواهیم تحلیل انجام دهیم و یا این که در یک Backlog قصد داریم کارها را به کارهای کوچکتری بشکنیم.
* **Bug:** زمانی که ایرادی را مشاهده و یا گزارشی از وجود یک باگ در محصول دریافت میکنیم، با ایجاد این تسک و وارد نمودن جزئیات مرتبط، جهت رفع آن اقدام می­نمائیم.
* **Customer:** زمانی مورد استفاده قرار می گیرد که می خواهیم بدانیم که به ازای یک Customer باید چه کارهایی انجام دهیم. به عنوان مثال چه محصولاتی از ما خریداری کرده است، ثبت نمودن اطلاعات وی و ... . در این جا می توانیم یک Work Item از نوع Customer ایجاد نماییم تا بتوانیم تشخیص دهیم که اگر یک Feature یا Product Backlog Item یا حتی یک Bug به سیستم اضافه شده است، توسط کدام Customer بوده است و ...
* **Epic :** فعالیت های بلند مدت را شامل میشود. مانند پروژه های کلان سازمانی
* **Feature :** فعالیت های مرتبط با پروژه ها که به زمان دو الی سه ماهه جهت تکمیل نیاز دارند.
* **Impediment :** بیان کننده موانعی است که در کیفیت، زمان بندی، ابعاد و به طور کلی در به سرانجام رساندن پروژه تأثیر گذار است.
* **Knowledge Acquistion:** برای کارهایی مورد استفاده قرار می‌گیرد که واقعا آزمون و خطا هستند (زمانی که ما مسیری را طی می‌نماییم تا مشخص شود که انتهای آن مسیر بن بست است یا خیر، که چنان چه انتهای آن مسیر به بن بست بخورد آن مسیر مناسب برای سازمان نمی‌باشد و می‌توانیم گزینه‌های درست را انتخاب نماییم.)
* **Product:** می‌توانیم محصول را کلان‌ترین لایه تعریف کرده و برای Product می‌توان Epic، Feature و ... را تعریف نمود. ( یک حالت درخت واره‌ای می‌شود از کارهایی که مانده و هنوز تکمیل نشده‌اند.)
* **Product Backlog Item**
* **Support:** برای کارهایی مورد استفاده قرار می‌گیرد که مشتری تماس می‌گیرد و مشکلاتی را پیگیری می‌نماید. حال این موارد چقدر زمان تیم را گرفته است، ممکن است زمان تیم پشتیبانی و تست را نیز گرفته باشد و ...
* **Task :** تمامی فعالیت ها در پروژه در نهایت به تسک های کوچکی با دوره های زمانی کوتاه جهت اجرا تقسیم میگردد.
* **Task Est:** معمولا تسک استقرار را از تسک‌های دیگر جدا می‌نماییم تا مشخص شود که آن‌ها مختص کارهایی هستند که به تیم پشتیبانی مربوط می‌شود و از آن جایی که تیم پشتیبانی دغدغه‌های بسیاری دارند، Work Item‌های آن فیلدهای دیگری نیاز دارد تا بتواند Data را از مشتری دریافت نموده و رکورد کند و بعد روی آن‌ها تصمیم‌گیری شود.
* **Technical Dept:** مجموعه مشکلات موجود در مسیر توسعه نرم افزار است که به پیشرفت ناپایدار ارزش مشتری دامن میزند.
* **Technical Work:** برای کارهای فنی مورد استفاده قرار می‌گیرد. مثل نصب SQL، تنظیم یک وب سرور که برای سازمان Business محسوب نمی‌شوند ولی مجبور به انجام آن‌ها هسیم.

با جمع‌آوری این ابعاد از کارهای غیرمرتبط خیلی راحت با یک گزارش می‌توان متوجه شد که اگر تیم It از نظر مدیریت سازمان کند کار می‌کند مشکل از کجاست؟ Technical Dept است و یا ... .

# فیلدهای جدید مورد نیاز به شرح زیر هستند :

نوع Product

فیلدهای Priority و Effort اضافه می‌گردد.

بعد از اضافه کردن Product مراحل زیر طی گردد:

Collection => Collection Setting => Process => Scrum => Create inherited process => تعیین نام (Scrum-indexمثلا ) => Set as default process => کلیک روی Scrum => projects => نام پروژه => کلیک روی Change Process => انتقال به Scrum-index

به این صورت مدل فرآیندی پروسه جاری تغییر می‌کند. به All Process برمی‌گردیم:

All process => کلیک روی Scrum-index => New Work Item Type =>تعیین نام Product => انتخاب ایکون و رنگ => Create

وقتی نوع جدیدی اضافه می‌نماییم، خالی می‌باشد. فقط Description و سه لینک مرتبط با Deployment را دارد:

1. **Deployment:** نشان می‌دهد که این Work Item در کدام Deployment‌ها یا CICD‌ها وجود دارد.
2. **Development:** نشان دهنده کدهایی است که برای آن نوشته شده است.
3. **Related Work:** نشان دهنده Work Item‌های دیگری است که به آن متصل می‌باشند.

حال در Product ایجاد شده:

کلیک روی Add a Field => Use An Existing Field => Priority => Add Field

New Field => Use An Existing Field => Effort => Add Field

این دو فیلد از قبل وجود دارند و خوبی آن این است که هنگام گزارش گیری در Work Item‌های مختلف فقط کافی است بگوییم آن Work Item‌هایی که Priority آن‌ها مثلا یک است یا دو یا ... و دیگر Type آن‌ها اهمیتی ندارد. ولی بعضی Field‌ها وجود ندارند، مثل Type of System که لازم است آن‌ها را ایجاد نماییم.

لایه Backlog Items

* به تمامی انواع این لایه فیلد Acceptance Criteria به عنوان Definition of Ready اضافه می‌گردد.
* به تمامی انواع این لایه فیلد Type of System از نوع Pick List و Effort و Priority اضافه می‌گردد.



چه **زمانی** Type of Systemرا اضافه نماییم؟

به عنوان مثال در نظر بگیرید می خواهید سیستمی را Handle نمایید، ما گفتیم که برای Handle نمودن ماژول‌های Product از Area استفاده می‌کنیم. حال فرض کنید که شما آن Business را می‌خواهید در پنل Admin خود وارد نمایید یا در IOS یا Android یا پنلی که وب سایت شرکت می‌باشد و یا Client Side (Client می‌بیند که دارد کار می‌کند) و ... مسئله‌ای که مطرح می‌شود این است که آن Business در کدام یک از این پنل‌ها برایش کار انجام می‌شود؟ آن جاست که فیلدی به نام Work Item اضافه می‌نماییم. مثل Type of System که می‌تواند Pick List باشد.

All Process => New Work Item Type => ساخت Technical Dept => تعیین رنگ و آیکون => Create => Add a field => ایجاد Type => Create a Field => تعیین نام Type Of System => تعیین Type (Pick List <string>) => تعیین مقادیرAdd Value

ذکر مثال‌هایی برای Add Value :

* **Android**
* **Admin Panel**
* **Client Side**
* **IOS**
* **None**



نوع Support

نوع Related Solution از نوع Pick List اضافه می‌گردد.



نوع Task

فیلد Type of System و Estimated Work از نوع دسیمال و Actual Work از نوع دسیمال اضافه می‌گردند. این نوع فیلد زمان در آن دخیل می باشد.

Task دارای فیلدی به نام Remaining Work می‌باشد که جزء فیلدهای بسار خاص است ولی متأسفانه به Functionality آن دسترسی نداریم. یعنی هر بار که مقداری را برای آن تنظیم می‌نمایید آن مقدار کم می‌شود و درنتیجه مقدار آن صفر می‌شود که درست آن به این صورت است که Remaining Work برای تسکی که Done شده صفر باشد ولی مشکلی که وجود دارد این است که همیشه تسکی که Done شده مقدارش صفر است. به عنوان مثال کاری را درنظر بگیرید که مشخص کردید آن کار را در 2 ساعت انجام دهید ولی 5 ساعته انجام شده است، باید علت بررسی شود تا در Sprint‌های بعدی این مشکل ایجاد نشود. به همین علت Remaining Work نمی‌تواند پاسخگوی جواب ما باشد، چون کلا صفر است زمانی که Done می‌شود. به همین علت دو فیلد به Task اضافه می‌نماییم.



نوع Task Est

فیلدهای Effort و Actual Work و Part of System اضافه گردد.



توضیح Acceptance Criteria

یعنی اگر اسکرام کار می‌کنیم و به عنوان Product Owner کاری را تعریف می‌نماییم و به تیم توسعه ارائه می‌دهیم حتما باید Acceptance Criteria یا هدف را مشخص نماییم. Acceptance Criteria هدف آن Business یا نیازمندی فنی می‌باشد.

درواقع Acceptance Criteria روی Produce Backlog Item ها صحبت می‌کند که اگر Produce Backlog Item یی قرار است تحویل دهیم، Product Owner دقیقا در چه شرایطی آن را از ما می پذیرد. ( چه اتفاقاتی باید بیفتد تا آن را از ما بپذیرد)



دیدگاه اسکرام

**در لایه Product Backlog Item**

* **Effort‌ها** همان Story Point‌ها هستند. یعنی آیا این کار از نظر ما خیلی سریع می‌تواند انجام شود، طول می‎‌کشد، خیلی طول می‌کشد، کار بسیار سنگین و زمان بر است و ...
* **Priority:** معمولا برای آن اولویت 1 تا 4 در نظر گرفته می‌شود و در قسمت‌های مختلفی خودش را نشان می‌دهد و برای مواقعی است که شاید دو کار به نسبت بسیار نزدیک به یکدیگرند یا می‌خواهیم در گزارشاتمان این تأثیر را مشاهده کنیم. پس معمولا 4 لایه آن را در نظر می‌گیریم. تنها استفاده‌ای که دارد در گزارشات است.
* **Business Value و Value Area:** در ذات، اسکرام نیستند و وارد جزئیات آن‌ها نمی‌شویم. استفاده از آن‌ها می‌تواند فقط در تهیه گزارشات به ما کمک کند.



در لایه Task

1. **Remaining Work:** همان طور که گفته شد از Remaining Work استفاده‌ای نمی‌شد.
2. **Activity:** از این فیلد زمانی استفاده می‌شود که تیم تحلیل و یا تست اضافه شود، در آن زمان این فیلد در تهیه گزارشات خیلی کمک می‌کند.
3. **Block:** جزء فیلدهایی است که گاهی اوقات مورد استفاده قرار می‌گیرد، زیرا بعضی از Task ها ممکن است Block شوند.



در لایه Bug

* **Priority و Severity:** بسیار مهم هستند و با یکدیگر ماتریسی به وجود می‌آورند که براساس آن ماتریس ما تصمیم می گیریم که کدام Bug ها را در Sprint جاری رفع نماییم و ... .
* **Effort:** همان Effort اسکرامی می‌باشد.
* **Remaining Work و Activity:** خیلی استفاده نمی‌شوند.
* **Build**

****

* برای این منظور که Bug مورد نظر در کدام Environment می باشد، بهتر است فیلدی اضافه گردد.
* کل اسکرام یک روش تجربی است. یعنی برنامه‌ریزی یک Plan برای دوسال آینده را کنار بگذارید و .... به عنوان مثال Google زمانی که می‌خواست Chrome را بسازد، در طی دو سال آن را ساختند که در ابتدا امکان Bookmark را برای آن در نظر نگرفتند و بیان کردند که کسی دیگر از Bookmark استفاده نمی کند و پس از Develop کردن، یک نسخه Beta درون سازمانی را ارائه دادند و مشاهده نمودند که بسیاری از افراد به Bookmark نیاز دارند و ... . پس همه این موارد باعث گشت که Google خیلی سریع موتور جستجویش را ارئه دهد.
* مشتری گاها حتی خودش هم نمی‌داند که چه چیزی را می‌خواهد، فرض کنید تحلیل بلند بالایی را انجام می‌دهیم، دو ماه بعد می‌خواهیم آن را پیاده‌سازی نماییم و به مشتری تحویل دهیم، مسلما درصد شکست پروژه بالا می‌رود. در اسکرام وظیفه درست کردن Backlog و ... برعهده تیم اسکرام می‌باشد و نقشه این گونه است که Product Owner به جای Customer و User صحبت می‌کند و Business را ارائه می‌دهد. پس باید در ابتدا نقش‌های اسکرام در سازمان مشخص شود.

**»Microsoft Azure DevOps (Session4) «**

مقدمه

در این سند آموزش نحوه پیاده‌سازی و نگهداری از بخش محصولات سازمان، اجایل و دانشنامه فنی سازمان را مورد بررسی قرار می‌دهیم.

مدیریت محصول در Azure DevOps

برای آن که بتوانیم محصول خود را در Azure DevOps مدیریت نماییم دو بخش ما را درگیر می‌نمایند:

1. **OverView**
2. **Board**

این دو با یکدیگر کمک می‌کنند تا بتوانیم نیازمندی‌های محصول را برطرف نماییم.



Board

این بخش بسته به آن که شما از چه پروسه‌ای استفاده می‌کنید، ظاهرش متفاوت می‌شود.

زیرمجموعه‌های Board

1. **Work Item:** نسبت به بقیه موارد به هم ریخته‌تر و شلوغ‌تر می‌باشد، و علت آن هم این است که در این قسمت شما همه Work Item‌ها که اهمیتی ندارد مختص چه تیم اسکرامی هستند یا متعلق به چه محصولی می‌باشند و ...، چنانچه شما Permission داشته باشید، آن‌ها را می‌بینید و به شما امکانات فیلترینگ بیشتری را اختصاص می‌دهد. نکته دیگر سطل زباله (Recycle Bin) آن است، که چنان چه Work Item یی را پاک کرده باشید تا 30 روز فرصت دارید آن را از سطل زباله برگردانید. دراصل یک ماهیت General دارد و توصیه می‌شود زیاد به آن کاری نداشته باشید.
2. **Board:** اسم آن کمی شما را دچار اشتباه می‌کند. منظور از Board همان KanBan است، یعنی چنانچه می‌خواهید به جای اسکرام از روش KanBan استفاده نمایید، می‌توانید در این بخش کلا به روش KanBan کار کنید.



متدولوژی KanBan

چنانچه تصور می‌کنید که سازمان شما تازه می‌خواهد از روش اسکرام استفاده نماید و این روش برای سازمان سنگین و پیچیده می‌باشد، یکی از روش‌هایی که می‌توانید استفاده کنید تا کم‌کم به اسکرام مهاجرت نمایید KanBan می‌باشد.

KanBan آن مسائل خاص، رویدادها (‌Eventها) و Data‌هایی مثل اسکرام را ندارد و تنها مسئله‌ای که وجود دارد محدود کردن کار در این Progress (WIT) می‌باشد.

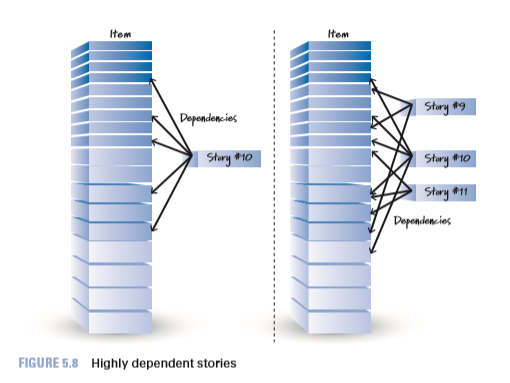
****

* **Approve:** از 0 تا 5 در نظر گرفته می‌شود و سقف آن به صورت Default روی 5 تنظیم می‌شود، زیرا چنان چه انسان در یک روز دو کار مختلف برای انجام داشته باشد بهتر می تواند بین آن ها Switch کند تا این که 5 کار مختلف را برای انجام داشته باشد، در آن زمان مدیریت آن‌ها دشوار خواهد شد.
* **Approved:** یعنی تأیید شده تا انجام شود و اکنون تیم توسعه روی انجام شدن آن زمان می‌گذارند. به راحتی می‌توانید با Drag And Drop هر کاری را (Bug و ...) داخل آن قرار دهید.
* **Commit:** یعنی روی آن پروژه کار می شود تا به انجام برسد.
* **Done:** یعنی کار به اتمام رسیده است.

به طور کلی KanBan بدون Effort، Sprint، Velocity و ... به سادگی امکان مدیریت کارها را فراهم می‌کند و برای بچه‌های پشتیبانی یا استقرار مناسب می‌باشد.( افرادی که برای آن‌ها بسیاری از کارها خیلی ناگهانی ایجاد می‌شوند و باید سریع انجام شوند.)

در قسمت New اگر روی ... کلیک نمایید و گزینه Add Test را بزنید، چنان چه از اسکرام استفاده می‌نمایید توقع Azure از شما این است که به ازای هر Sprint یک Test Plan داشته باشید ولی در KanBan چون time و sprint و ... وجود ندارند، ما Add Test را در همان Level اول انجام می‌دهیم.

1. **BackLog:** اگر می‌خواهیم از روش اسکرام استفاده نماییم، این بخش این امکان را به ما می‌دهد. BackLogدرواقع متعلق به تمام محصول می‌شود که هرچقدر بالای محصولات باشیم Item‌ها و Story‌ها شکسته شده‌اند و جزئیات آن‌ها مشخص شده است. ولی آن BackLog کل محصول می‌باشد. به طور کلی قسمت‌های پایین‌تر یک Epic و Feature هستند که هنوز خرد نشده‌اند تا جزئیات آن‌ها مشخص شود و متوجه شویم که چه کاری باید انجام دهیم. درواقع آن‌هایی که بالاتر قرار دارند کارهای خرد شده‌ای هستند که Detail آن‌ها مشخص شده و هر چه رو به پایین حرکت کنیم، حجم اطلاعات بزرگتر می‌شود.



1. **Sprints:** درواقع روی BackLog متعلق به خود Sprint کار می‌کند. یعنی این امکان را به شما می‌دهد تا محصول را دسته‌بندی کنید و پس از دسته‌بندی می‌توانید کارها را در سطح کل محصول روی BackLog تفکیک نمایید. حال Azure می‌بایست این امکان را فراهم نماید تا بتوانید به بهترین شکل ممکن این Sprint‌ها را اجرا و مدیریت نمایید. درواقع این بخش روی یک Sprint متمرکز می شود.
2. **Queries:** فرض کنید می‌خواهید گزارش تهیه نمایید، یا مدیر از شما گزارش خاصی را خواسته است و ... آن جا نیاز به نوشتن Queries می‌باشد تا بتوانیم از بین Data‌ها اطلاعات مورد نظرمان را نمایش دهیم.



Type Of Queries

* **Flat List of Work Items:** فقط از این نوع می‌توان چارت و دیاگرام تهیه نمود.
* **Work Items and Direct Links**
* **Tree of Work Items**

1. **Analytics Views**



بررسی یک Sample برای Board

محصولی به اسم Unique Shoes را در نظر بگیرید که یک فروشگاه اینترنتی می‌باشد و به مشتری این امکان را می‌دهد تا از این سایت استفاده نماید و کفشی را که مد نظرش است را Custom کند.

با استفاده از Brows Marketplace به سایت Marketplace رفته و Extention مورد نظرتان را نصب نمایید. توصیه می‌شود از Extention‌هایی استفاده نمایید که توسط Microsoft ارائه شده‌اند، زیرا جواب خود را پس داده‌اند.



Extention Delivery Plans چه امکانتی را فراهم می‌کند؟

درواقع مشکل BackLog این است که Sprint‌ها را به تفکیک نشان نمی‌دهد و مشکل Sprint این است که خیلی روی Sprint‌ها Zoom می‌کند و بر روی آن دو هفته و یک هفته Sprint متمرکز می شود، پس جای یک چیزی که عقب تر بیاید و بررسی کند که Sprint به Sprint کنار هم کارها چگونه Deliver می‌شوند، خالی می‌باشد. Product Role Map یعنی در لایه‌های مختلف PBI یا Epic چه اتفاقی می‌افتد و ... .

حال در ابتدا باید Area را برای محصول UniqShoe مشخص نماییم:

Project Settings => Project Configuration => Area

در Interations تاریخ Sprint‌ها را مشخص می‌نماییم و سپس وارد BackLog می‌شویم.

کلیک روی Column Options => Add Area Path

همچنین Tags کاربرد چندانی ندارد و آن را حذف نمایید. اکنون مشخص است در کدام Sprint و روی چه کارهایی زمان می‌گذاریم. همان طور که گفته شد کاری که در بالاترین سطح قرار دارد، باید برایش برنامه ریزی نماییم. پس برای آن هایی که هنوز Sprint‌ها مشخص نشده است، Sprint را مشخص نمایید.

BackLog‌ها فقط کارهایی را نمایش می‌دهند که باید انجام شوند، یعنی چنان چه وضعیت آن‌ها Done شده باشد، دیگر آن‌ها را نشان نمی‌دهد. زیرا وظیفه BackLog نمایش کارهایی است که باقی مانده‌اند و می بایست آن‌ها را انجام دهید.

حال می‌توانیم سه مورد زیر را در BackLog تعریف نماییم:

1. **Epic**
2. **Feature**
3. **Backlog Items**

ابتدا یک Product ایجاد کرده و به آن یک Epic اضافه می‌نماییم:

Unique Shoes (Product) => Buy Shoes(Epic) یعنی این فروشگاه قابلیت خرید کفش را مهیا می‌کند.

سپس رو + کنار Epic کلیک کرده و Feature را اضافه می‌نماییم. یعنی یک ساختار سلسله مراتبی ( درختی ) ارائه می‌دهد.

Search Shoes

View Search Result

Unique Shoes => Buy Shoes =>

پس به هر Feature می‌توانید Item اضافه نمایید(PBI):

One Man Shoe

One Woman Shoe

List Top 20 Products کاربر بتواند 20 محصول برتر را لیست نماید.



مفهوم User Story

به معنای داستان کاربر می‌باشد. یعنی کاربر قرار است چگونه از سیستم استفاده نماید. در اصل PBI همان User Story می‌باشد.

اگر روی PBI کلیک کنیم دو گزینه مهم وجود دارند:

1. **Description:** همیشه منظور این است as a persona. به عنوان مثال در این پروژه خانمی دوست دارد خرید کند ولی هر کفشی را نمی‌پسندد و علاقه دارد بتواند کفش خود را Customize کند تا متفاوت باشد و همچنین به خرید با تخفیف نیز علاقه‌مند است. به همین علت همیشه در Description باید بنویسیم as a کدام Persona. یعنی چه چیزی می‌خواهم و چرا به آن نیاز دارم( می‌توان مشخص کرد که چه Person‌هایی با این PBI در ارتباط هستند.)

مثال: من به عنوان حسین، نیاز دارم که 20 محصول برتر را بتوانم در نتایج جستجو نمایم، زیرا نظر سایر مشتریان سایت برای من دارای اهمیت است.

1. **Acceptance Criteria:** در این قسمت باید تجربه آن کاربر نوشته شود (روند و نحوه انجام کار شرح داده شود.) به عنوان مثال برای List Top 20 Products می‌خواهیم وقتی Login کردیم یک گزینه هنگام سرچ نمایش داده شود که می‌خواهید لیست 20 محصول برتر را مشاهده نمایید یا خیر و ... . اگر این قسمت را دقیق ننویسیم به هزار روش می‌توان آن پروژه را طراحی و پیاده سازی نمود، پس لازم است نحوه پیاده سازی را مشخص نماییم.

مثال: اگر کاربر Login کرده باشد و یا خیر فرآیند زیر صورت خواهد گرفت:

زمانی که کاربر به صفحه جستجو مراجعه نماید در زیر دکمه سرچ یک تاگل باتن وجود خواهد داشت که با فعال‌سازی آن نتایج جستجو با توجه به 20 محصول پربازدید اولویت‌بندی خواهد شد.



* Product باید آن بلوک‌های بزرگ را مطرح کند و همه بتوانند روی آن صحبت نمایند تا به بلوک‌های کوچکتری تبدیل شوند و جزئیات آن‌ها مشخص شود. این کار وظیفه Product Owner می‌باشد ولی بحث را باز می‌گذارد (پس باید هم FeedBack مشتری را دریافت کند و هم FeedBack بچه‌های تیم را.)
* Acceptance برای پر کردن جای خالی Use Case‌ها (UML) می‌باشد تا به توسعه دهنده بفهمانیم که کاربر دقیقا چه می‌خواهد.
* حال اگر در Level PBI ما Parent را on نماییم می‌توانیم زیرساخت‌ها را مشاهده نماییم.
* می توانیم چند Work Item را با یکدیگر Edit نماییم به شرط آن که برای آن‌ها Field مشترک داشته باشیم. مثل Area Path
* Edit => Field (Area Path) => Value (SoftDept/Product)
* چنانچه PBI‌ها را به شکل زیر تقسیم نماییم به راحتی می‌توانیم گزارش تهیه کنیم:
  + مثلا برای کدام PBI زمان بیشتری صرف شده است یا کدام PBI دارای Bug بیشتری می‌باشد و ...
  + حال در قسمت Project Configuration به Area تعریف شده Child اضافه می‌نماییم و در BackLog قسمتArea Path را به روزرسانی می‌کنیم.
* Option دیگری که در BackLog موجود است Forecasting است. ابتدا Effort‌ها را وارد می‌نماییم، حال Product owner می بیند که اگر Forecasting را 10 قرار دهید خود Azure بر اساس Effort‌هایی که وارد کرده اید، مشخص می کند که در هر Sprint چه کارهایی را می‌توانید انجام دهید (مجموع Effor‌ها 10 شود بقیه کارها به Sprint بعدی منتقل می‌شود). حال سعی کنید براساس اولویت کارها را در PBI تنظیم نمایید.
* از دیگر Option‌های موجود در BackLog یکی Mapping است و دیگری Planning .



**Mapping**

* می توانید لایه‌ی بالاتر را مشخص نمایید (Parent) و حتی در صورت اشتباه بودن آن را عوض نمایید.
* حال اگر برای PBI‌ها Task ایجاد نماییم و در حالت Done قرار دهیم، بازهم آن‌ها را نمایش می‌دهد. اگر PBI‌ها Done شوند دیگر نمایش داده نمی‌شوند.
* **Completed Child Items:** اگر off شود Child‌هایی که Done شده باشند را دیگر نشان نمی‌دهد.
* این امکانات را کسی می‌تواند تغییر دهد که Admin تیم اسکرام باشد.



نکاتی در خصوص KanBan

* کسی می‌تواند Config آن را تغییر دهد که Admin تیم اسکرام باشد.
* روی چرخ دنده کلیک کنید و ساختار را Customize نمایید:
* **Fields:** می‌توانید تنظیم کنید که چه فیلدهایی نمایش داده شوند و چه فیلدهای نمایش داده نشوند.
* **Style:** می‌توانید Style تعریف کنید که در چه شرایطی بک گراند کدام یک چه رنگی شود و ... .
* **Tag Colors:** می توانید Tag بزنید و براساس Tag‌ها رنگ تعیین کنید.
* **Annotations:** تعیین فعال بودن یا نبودن
* **Columns:** قاعدتا باید براساس State‌های موجود در Work Item‌ها باشند. اگر ستون‌هایی اضافه کنید که با State‌های موجود برابر نباشد بعدا اگر خواستید KanBan را کنار بگذارید و به اسکرام مهاجرت کنید، دچار مشکل می‌شوید.
* **Swimlanes:** اگر KanBan کار می‌کنید ممکن است Swimlanes داشته باشید (کارهای پر اولویت). مثلا یک Bug که نسبت به بقیه اولویت بالاتری دارد.

محاسبه می‌کند چه میزان از کارها انجام شده و چه میزان باقی مانده

Column‌های معمولی را در Work Item اضافه می‌کند.

Add a Column

Process Bar

میزان انجام شدن کارها را مشخص می‌کند.

Add a Rollup Column

Total Number

**«Microsoft Azure Devops (session 5)»**

# 

# 

مقدمه

در این جلسه ساختار پروژه های ایجاد شده در BackLog و Query ها و Dashboard ها را مورد بررسی قرار خواهیم داد.



بررسی ساختار پروژه ها در BackLog

اگر Area های هر Product مشخص نشود، در آینده در گرفتن کوئری ها دچار مشکل خواهید شد، زیرا نمی توان مشخص کرد که هر قسمت مربوط به کدام محصول می باشد.

* در تعریف پروژه ها سعی کنید پروژه هایی که به صورت مجزا یک عمل را انجام می دهند ( همانند پروژه مدیریت تسک ها Job Scheduler) که می تواند هر پروژه ای را اجرا کرده و مدیریت نماید، به صورت مستقل در نظر بگیرید.
* جهت تغییر نوع Work Item تعریف شده به صورت زیر عمل نمایید:
* کلیک روی Work Item ( مثل PBI ) => کلیک روی ... => Change Type => Type => تغییر نوع
* برای اضافه کردن فیلد Acceptance Criteria و مانند آن به Product یا سایر Work Item ها به صورت زیر عمل نمایید:
* به بالاترین لایه بروید ( Index Collection ) => Settings Collection=> Process => Scrum-Index => Product => New Field => Acceptance Criteria را اضافه نمایید.
* اگر نوع یک PBI را به Product تغییر دهیم، با وجود آن که باید در بالاترین لایه قرار بگیرد ( چون Product است )، ولی به علت آن که Child یک Feature می باشد، در زیر مجموعه آن قرار می گیرد.
* ساختار تعریف BackLog ها به صورتی است که ابتدا می بایست Product را مشخص کرده و سپس Feature هایی که می خواهیم برای آن محصول Develop نماییم را مشخص می نماییم و در ادامه Feature را به یکسری PBI شکسته و Task ها و سایر موارد، در PBI ها تعریف می شوند.
* در Description معمولا بنویسید که چه کسی محصول را می خواهد، چرا می خواهد و چه مشکلی را حل می نماید. در Acceptance Criteria نیز بنویسید که وقتی این Business انجام شود، چه کارهایی می تواند انجام دهد.
* در ساختار درختی ترتیب حائز اهمیت می باشد و نشان دهنده اولویت انجام کارها می باشد.
* PBI ها کارهای فنی ای هستند که می بایست انجام شوند و هر PBI باید در یک Sprint انجام شود.
* با اضافه کردن Iteration Path می توانید مشاهده نمایید که هر Work Item در کدام Sprint تعریف شده است.
* مشخص کردن Task ها بر عهده اعضای تیم است که بر اساس PBI ها، باید Task های خود را مشخص کرده و میزان زمان انجام هر Task را نیز مشخص کنند.

دیاگرام های Velocity و Cumulative Flow Diagram

* در BackLog گزینه ای به نام Analytics وجود دارد که دو نوع دیاگرام Velocity و Cumulative Flow Diagram را برای گزارش گیری ارائه می دهند. کار این دو دیاگرام نشان دادن تعداد کارهای انجام شده و مدت زمان انجام شدن آن ها می باشد.
* نمودار Velocity گزارشی از کارهای انجام شده یا نشده و ... را نشان می دهد و ما می توانیم بر اساس این نمودار علت انجام نشدن برخی از کارها را بررسی نماییم که چرا تیم نتوانشته کار تعریف شده را به پایان برساند.
* نمودار Cumulative Flow Diagram ( CFD ) به شما نشان می دهد که چه میزان از کارها انجام شده است و چه میزان از آن ها در حال برنامه ریزی می باشد. در این نمودار رنگ ها تعیین کننده هستند.
* این دیاگرام ها روی Sprint تمرکز نمی کنند و بسیار کلی می باشند.

بررسی Sprint ها

* در Sprint ها نیز Analytics وجود دارد که این نمودار در این قسمت، جزئیات بیشتری را نشان می دهد و در هر Sprint می توان بازه زمانی را نیز مشخص نمود.
* منوی دیگری که در Sprint موجود است، Task Board می باشد. شخص در این قسمت می تواند مشخص کند که تمامی کارهایی که به وی اختصاص داده شده است را به صورت یک Board مشاهده کرده و مدیریت نماید.
* منوی دیگری به اسم Capacity نیز وجود دارد که مشخص می کند تیم تعریف شده از چه افرادی تشکیل شده است و این تیم هر کدام چه میزان زمان می توانند روی کارهای مختلف بگذارند. حال اگر به منوی BackLog در Sprint رفته و گزینه Work Details را فعال نمایید، می توانید جزئیات را مشاهده کنید. جزئیات موجود در Work Details از روی Remaining Work های درون Task مشخص می شود.
* Burn Down در Sprint: در این نمودار شما می توانید مشخص نمایید که از کدام اسپرینت گزارش می خواهید و اسپرینت مورد نظر را مشخص نمایید.

# 

بررسی Queryها

گزینه دیگر Query ها هستند که به دو دسته تقسیم می شوند:

1. My Queries : فقط خود شما آن را می بینید و هیچ راهی برای نمایش آن وجود ندارد.
2. Share Queries : در این قسمت می توانید کوئری ها را با سایر همکاران به اشتراک بگذارید. در این قسمت برای نظم و دسترسی راحت تر، ساختار پوشه بندی ایجاد کنید.

* در این قسمت می توانید یک کوئری جدید ایجاد کنید و آن را اجرا نمایید. درواقع می توانید روی تمام Work Item های موجود کوئری ایجاد نمایید و Work Item های خاصی را نمایش دهید.
* اگر در کوئری ها نوع فیلد را Area Path در نظر بگیرید، در آن هنگام می بایست Operation آن را Under در نظر بگیرید و این به دلیل ساختار درختی Area ها می باشد.
* می بایست Area Path را از طریق Column Options اضافه کنید تا در روزهای آتی دچار مشکل نشوید، زیرا می خواهید از روی کوئری ایجاد شده Chart تهیه کرده و آن را در داشبورد اضافه نمایید.
* چک باکس کوچکی در بالای صفحه قرار دارد ( Query Across Projects )، با فعال کردن این گزینه می توانید از تمام Collection های تعریف شده کوئری بگیرید، اگر دسترسی شما Full باشد.
* گزینه دیگر Flat List of Work Items می باشد که می توانید از آن ها Chart تهیه نمایید. هم چنین از دو گزینه بعدی آن یعنی Work Items and Direct Links و Tree of Work Item می توانید گزارش تهیه نمایید ولی امکان استفاده از آن در Chart ها و دیاگرام ها نمی باشد.
* در کوئری ها اگر مقدار Field را Iteration path در نظر بگیرید، می توانید مقدار آن را @Current Iteration ( اسپرینت جاری ) تعیین کهWork Item های اسپرینت جاری را نمایش دهید.
* در کوئری ها اگر بخواهید دو Work Item را لینک نمایید از Work Items and Direct Link یا Tree of Work Items استفاده می شود. فقط لازم است بدانید که از این دو حالت نمی توانید Chart تهیه نمایید و فقط یک ساختار درخت واره ای را به شما ارائه می دهد.
* در کوئری ها می توانید امنیت و محدودیت ایجاد نمایید که چه افرادی بتوانند آن را مشاهده نمایند یا ویرایش کنند. برای این منظور روی سه نقطه کنار کوئری کلیک کرده و روی Security کلیک نمایید. اگر کسی دسترسی به Edit کردن یک کوئری را نداشته باشد در داشبورد هم نمی تواند این عمل را انجام دهد. هم چنین اگر دسترسی فردی را روی Read قرار دهید، در نتیجه فقط می تواند کوئری ها را ببیند و امکان تغییر و Edit را ندارد.



نحوه نوشتن شرط در Query

* به عنوان مثال می خواهیم تمام باگ هایی که اولویت 1 را دارند روی یک محصول و تمام باگ هایی که اولویت 2 را دارند، روی محصول دیگری نمایش دهیم. حال می خواهیم نتیجه این 2 if را که هرکدام در یک پرانتز قرار دارند و بین آن ها یک Or می باشد، را در یک کوئری ببینیم. برای نوشتن این کوئری می بایست نکات زیر را در نظر بگیرید.
* برای ایجاد پرانتز بین if ها باید آن ها را Group نمایید. به مثال زیر توجه نمایید:

Work Item Type = Product BackLog Item

AND Area Path Under SoftDept

OR Work Item Type = Task

AND Area Path Under SoftDept / SanaAPI

کوئری بالا به صورت زیر تعریف می شود:

( Work Item Type = Product BackLog Item AND Area Path Under SoftDept ) OR ( Work Item Type = Task AND Area Path Under SoftDept / SanaAPI )

* هر تغییری که در کوئری ایجاد می نمایید، باید Save شده تا در Chart قابل نمایش باشد. Chart را می توانید ایجاد کرده و به داشبورد منتقل نمایید.
* نمودارهای مختلفی را می توانید در قسمت Chart ایجاد نمایید. برای اضافه نمودن فیلدهای عددی مثل Effort به Chart می بایست از قسمت Column Options آن را اضافه نموده تا در Chart بتوانید از آن استفاده نمایید.
* بعضی از دیاگرام ها در قسمت Chart به شما امکانات بیشتری را ارائه می دهند. برای نمونه Pivot Chart به شما ستون اضافه ای را ارائه می دهد.
* پس از ایجاد Chart مورد نظر می توانید روی سه نقطه کنار آن کلیک کرده و آن را به داشبورد مورد نظر اضافه نمایید.

 بررسی Dashboard

* به ازای هر تیم اسکرام می توانید هر تعدادی که می خواهید داشبورد ایجاد نمایید که دسترسی کامل به داشبوردها را Administrator دارد.
* در Project Settings گزینه ای به اسم Dashboard وجود دارد که تیم اسکرام را مشخص می کند. در این قسمت می توانید محدودیت هایی برای اعضای تیم ایجاد نمایید.
* در داشبورد اگر روی چرخ دنده ای که وجود دارد کلیک نمایید، گزینه ای وجود دارد که با فعال کردن آن هر پنج دقیقه یکبار داشبورد به روز می شود.
* در داشبورد اگر روی Drop Down بالای صفحه کلیک نمایید ( مثلا SoftDept Team-Over View )، می توانید یک داشبورد جدید ایجاد کنید. در این قسمت می توانید مشخص کنید که داشبورد متعلق به تیم اسکرام یا کل تیم پروژه است. هم چنین می توانید مشخص نمایید که هر چند دقیقه یکبار به روز شود.
* داشبورد وضعیت کل پروژه را نشان می دهد و برای Customize کردن آن روی Edit کلیک کنید و برای تغییر سایز و سایر مشخصات نمودار روی ... کلیک کرده و Configure را انتخاب نمایید. در این قسمت می توانید طول و عرض نمودار را مشخص نمایید. در Configure هم چنین می توانید Query، Chart Type و ... را نیز Customize کنید.
* وقتی در کوئری ها یک Chart ایجاد می کنید و در داشبورد Add می نمایید، در حقیقت یک کپی از Chart را در داشبورد Add کرده اید و هر تغییری که در هر کدام ایجاد کنید در دیگری اعمال نمی شود.
* در داشبورد اگر روی Edit کلیک نمایید، لیستی از Widget ها را مشاهده می کنید که می توانید به سیستم اضافه کنید.



بررسی Widget ها

* Burn Down : نشان می دهد که به ازای اسپرینت و محصول، دقیقا چه اتفاقی در حال رخ دادن است، چه میزان کار داشتید و کارها در حال افزایش هستند یا کاهش و ... . گاهی مواقع تیم با وجود آن که کارها را به خوبی انجام می دهد، ولی باز هم تا پایان اسپرینت موفق به تحویل کامل محصول نمی شود و علت آن هم به دلیل آمدن کارهای بسیار زیاد سمت تیم می باشد. هنر این نمودار نشان دادن این موارد می باشد.
* Burn Up : بر عکس Burn Down این نمودار هر چه حجم کارها بیشتر می شود، سطح نمودار از کف بالاتر می رود. یعنی به جای آن که نشان دهیم که چقدر کار باقی مانده که انجام دهیم، می گوییم چه میزان کار انجام داده ایم. این نمودار فقط در داشبورد وجود دارد ولی Burn Down در اسپرینت نیز وجود دارد.
* Cycle Time : وظیفه اش نشان دادن پیش رفتن یا نرفتن کارها طبق Planing برنامه ریزی شده می باشد. مثلا یک PBI در آن بازه ی زمانی ای که مشخص شده است به پایان رسیده است یا خیلی زمان برده است و این مسئله در این نمودار مشخص است. در اصل نشان می دهد که از زمانی که کار تعریف شده تا زمانی که انجام شده است، چقدر زمان برده است.
* Velocity : در BackLog نیز وجود دارد، این نمودار اطلاعات اسپرینت های مختلف را نشان می دهد، هم چنین نشان می دهد که در هر اسپرینت چه میزان کار Done شده است.



نکات

* Summery نیز متعلق به کل تیم پروژه است.
* Search در Azure به صورتی است که در سه قسمت عملیات Search را انجام می دهد :

1. Code
2. Wiki
3. Work Item

**»Microsoft Azure DevOps (Session 6) «**



مقدمه

در جلسه قبل Backlog‌های ایجاد شده در Board، متناسب با نیاز سازمان و پروژه‌ها مورد بررسی قرار گرفت. یعنی ما ساختار کلی را به صورت زیر تعریف نمودیم و این ساختار مورد بررسی قرار گرفت:

Product

Epic

(1…n) Feature

(1…n)Pbi

(1…n)Task

به طور عمومی Pbi‌ها باید به Pbi‌های ریزتر شکسته شوند و ریز مسائل در Acceptance Criteria مطرح گردد. همچنین Effort‌ها نیز به Effort‌های ریزتر (مثلا 5، 3 و ...) تبدیل گردند.

در گام‌های بعدی این سؤال مطرح می‌شود که اولویت‌بندی این موارد به چه صورت می‌تواند باشد؟ به عنوان مثال یک اولویت‌بندی به نام CDP وجود دارد که اولویت‌بندی براساس هزینه می‌باشدکه مثلا این آیتم چند درصد برای سازمان مفید است و ... . این مورد حداکثر چیزی است که می توان اولویت‌بندی براساس هزینه و درآمد را کنترل نمود، که این Effort‌ها را می‌توان براساس آن‌ها تنظیم کرد تا سازمان بیشترین سود را ببرد.



بررسی WIKI

در این جلسه می‌خواهیم مفهوم Wiki را مورد بررسی قرار دهیم. می‌خواهیم بررسی کنیم که Wiki چگونه می‌تواند به سازمان کمک کند. به طور کلی Board‌ها خوب هستند ولی زمانی که شما می‌خواهید یک دید کلی نسبت به پروژه‌ها یا یک صفحه Landing داشته باشید، اسکرام این دید کلی را به شما نمی دهد که بتوانید نقطه شروعی داشته باشید و به کسی که تازه وارد سازمان شده است خط دهید و بفهمانید که از کجا اطلاعات پروژه را چک کند.

به عنوان مثال تصور کنید که یک پروژه سازمان Out Source شده و هیچ Bug و ... در Azure برای آن ثبت نشده است و از طرف دیگر هیچ شناسنامه و سندی هم ندارد. حال برای این مواقع دو کار زیر پیشنهاد می‌شود:

1. اگر قرار است به شخص جدیدی این پروژه را تحویل دهید برای آن که این روند ادامه پیدا نکند، شما باید ساختار را تغییر دهید، سازمان باید Agile کار کند و محصول یک Backlog‌یی پیدا کند.
2. از طرف دیگر هم در Knowledge Management برای آن شناسنامه‌ای درست کنید و توضیحاتی (ویس، ویدیو، متن و ...) تهیه نمایید. در قدیم از Share Point به این منظور استفاده می نمودند ولی Microsoft آن را از Azure جدا نمود.

در اصل Wiki‌ها یک ساختار درختی دارند، برای نمونه به سایت Azure DevOps Docs بروید و نمونه‌ای را مشاهده نمایید. درواقع محصول ما یک محصول نرم‌افزاری است که در Wiki در مورد آن صحبت می‌کنیم.

از Wiki استفاده می‌نمایند که مشخص کنند که به عنوان مثال یک توسعه‌دهنده چه وظایفی برعهده دارد؟ چگونه باید کار کند؟ در چه شرایطی کار شما را قبول می‌کنند؟ و ...

Docs GitLab نمونه جالبی برای Wiki می‌باشد، آن را بررسی نمایید.

به ازای هر Team Project تنها یک Wiki وجود دارد و Wiki متعلق به یک محصول خاص یا یک تیم اسکرام نمی‌باشد. پس به ازای هر Team Project یک Wiki خواهیم داشت.

مبحث دیگری که اهمیت دارد دانستن این مورد است که در پشت ‌Wikiها یک Git Repository قرار دارد و درنتیجه تمام قوانین Git Repository را شامل می‌شود و هر تغییری که در آن ایجاد می‌کنید مانند آن است که یک Commit و Push را انجام داده‌اید و این دو با هم اتفاق می‌افتند.

کلیک روی ... => کلیک روی Clone Wiki => مشخص شدن مسیر Git Repository

این Git Repository کنار بقیه Git Repository‌ها قابل مشاهده نیست و Hidden می‌باشد مگر آن که آدرس آن را در مرورگر وارد نمایید. Wiki حتی می‌تواند Branch داشته باشد. برای مرتب‌سازی در اولین قدم ما به یک ساختار نیازمندیم.

Wiki به صورت Mark Down کار می‌کند که معمولا در فایل های README.md این ساختار قابل مشاهده است.

برای تعیین ساختار یک سازمان در Wiki می‌توان مواردی مثل بحث تحقیق و توسعه، واحد فروش، نرم افزارهای مختلف، آموزش و ... را دسته بندی نمود.



ایجاد WIKI

به ازای هر پروژه ای که دارید یک صفحه در Wiki ایجاد کنید و به صورت کلی سناریو و گام‌های عملیاتی و مراحل مختلف پروژه را شرح دهید و یک ساختار درختی برای درک بهتر پروژه ایجاد نمایید و به Work Item‌های مربوطه رفرنس دهید.

در Wiki می‌توانید از تمام Tag‌های HTML و استایل‌های inline استفاده نمایید. لازم است نحوه کار با Mark Down را بدانید. مثلا نحوه ایجاد TOC و ... .

اگر Team Project به ازای سازمان باشد، صفحه اصلی را به شرح کلی پروژه می‌پردازیم و اگر به ازای یک پروژه یا Software Dept باشد، توضیحات آن را می‌نویسیم و پس از آن یک Over View مشخص کرده و در آن Subject‌ها (مثلا آموزش و ...) را مشخص می‌کنیم.

در ادامه ساختار درختی را برای Wiki ایجاد نمایید. حال اگر وارد Git Repository آن Wiki شوید، شکل گرفتن ساختار درختی آن Wiki را مشاهده خواهید نمود. ولی بعضی از آیتم‌ها هم فولدر و هم فایل .md را دارا می‌باشند و بعضی دیگر فقط فایل .md را دارا می‌باشند و این مورد به این دلیل است که آیتم‌هایی که Child دارند پس حتما می‌بایست فولدری برای نگهداری Child آن‌ها وجود داشته باشد.

View Revision تعداد نسخه‌ها و تغییراتی که در هر صفحه وجود دارد را نشان می‌دهد. درنتیجه شما تمام نسخه‌های Wiki را دارید.

****

* **#:** برای Work Item‌ها می باشد.
* **@:** برای مشخص کردن اشخاص استفاده می‌شود. پس از آن یک ایمیل برای فرد ارسال می‌شود.
* **!:** برای Pull Request‌هایی که هنوز در Wiki اعمال نشده‌اند.
* به طور خلاصه در Wiki محصول را شرح داده و جزئیات را بیان می‌نماییم و به Work Item مربوطه Assign می‌کنیم.



بررسی مفهوم Query Result

حال کمی فراتر می‌رویم و در Wiki روی ... کلیک کرده و Query Result ایجاد می‌نماییم. درواقع Query Result نتایج یک Query را به ما نشان می‌دهد.



معمولا می توان Wiki را در سه لایه ایجاد نمود

1. **لایه اول:** که به شما خط می دهد و به صورت کلی در مورد محصول صحبت می کند ( بیان Details از وظایف این لایه نمی باشد.)
2. **لایه دوم (OnBoardin):** این لایه کمی وارد Detail می‌شود و در مورد جزئیات محصول صحبت می‌کند.
3. **لایه سوم:** در این لایه Option‌ها و ماژول‌های استفاده شده در محصول را تعریف می‌نماییم و درمورد آن‌ها توضیح می‌دهیم.

در Wiki برای نظم کلی عکس‌ها و ... بهتر است عکس‌ها را که در پوشه‌ای به نام Attachments قرار دارند، پوشه‌بندی نمایید. اگر شما عکس‌ها را از طریق Drag and Drop داخل Wiki قرار دهید، به صورت نامرتبی قرار می‌گیرند. بنابراین آن‌ها را در Attachments پوشه‌بندی کرده و ساختار منظمی را برای آن‌ها ایجاد نمایید و درنهایت از آن آدرس برای نمایش عکس در Wiki استفاده کنید.

Wiki در یک لحظه می‌تواند Branch فعال را نشان دهد و قادر به نمایش چندین Branch با یکدیگر نمی‌باشد.

مسئله دیگری در Cross-Repo-Policy وجود دارد که باید به آن توجه داشت: یکسری از Policy‌ها را می‌توان روی کل Repository اعمال نمود که این موارد روی کل Git Repository اعمال می‌شود، درنتیجه اگر شما روی Cross-Repo-Policy یک Policy ایجاد نمایید، در آن صورت متأسفانه گاهی موارد در Wiki درگیر آن Policy هم می‌شوید که خطا می‌دهد که این Branch را به صورت دستی نمی توانید Update نمایید و باید Pull Request بزنید که این Wiki را برای شما Update کند.



مبحث REPO

ترتیبی وجود دارد که ابتدا Wiki محصول را مشخص کرده و سپس Board آن محصول را مشخص می‌نماییم و در مرحله سوم که Repo می‌باشد می‌خواهیم برای آن محصول کد بنویسیم. نکته‌ای که وجود دارد متصل کردن Repo و فرآیندهای داخل آن به بخش Board است. یعنی ما چنان چه کدی می‌نویسیم، آن کد مربوط به کدام Task است یا Branch مربوط به کدام Pbi، Technical Dept و ... می‌باشد.

Repo برای برطرف کردن نیازها شش آیتم دارد که پیچیده‌ترین آن‌ها بحث Pull Request و Branch‌ها می‌باشد.

1. **Files:** وظیفه آن نشان دادن این مورد است که چه فایل‌هایی وجود دارد و روی چه Branch‌هایی کار می‌کنیم و یا آن Branch چه فایل‌هایی دارد، History آن را نمایش می‌دهد و ... .

برای نامگذاری Repository‌ها سعی کنید نام آن‌ها چند سیلابه باشد. این مورد در جستجو کمک می‌نماید. به عنوان مثال:

Product اسم محصول-زیرسیستم آن

1. **Commits**
2. **Pushes**
3. **Branches:** Branch را باید محدود استفاده نمایید و Tag‌ها را باید بیشتر استفاده کنید. درواقع به جای نگه‌داشتن Branch باید Tag ایجاد نمایید. Branch و Pull Request به هم وابسته‌اند و Pull Request چیزی جز Merge دو Branch نمی‌باشد.
4. **Tags:** مثل یک پرچم است و بسیار حائز اهمیت می‌باشد. ما از کجا بدانیم تا کدام Commit ما ورژن 1 بوده است؟ برای همین منظور از Tag‌ها استفاده می‌نماییم.
5. **Pull Requests**

دو فایل مهم که هر Repository در ابتدا باید داشته باشند:

* **فایل README.md:** شناسنامه محصول در Git Repository می‌باشد.
* **.gitignore:** مشخص می‌کند چه فایل‌هایی نباید در Git Repository نمایش داده شوند.

فرآیند انتقال Git Repository به Repo در Azure

ابتدا از Gogs یک Repository را روی سیستم خود Clone نمایید و سپس توسط Command زیر Origin آن مشخص می شود:

git remote –v

حال خیلی راحت می توان Origin آن را عوض کرد و به Azure نسبت داد. برای این منظور از Command زیر استفاده نمایید. فقط نیاز است که Git Repo در Azure خالی باشد و History نداشته باشد. در ابتدا توسط Command زیر Remoteجاری را حذف نمایید:

git remote remove origin

حال گزینه دوم را از Git Repo کپی کنید و در Command قرار دهید تا Git Repository به Azure منتقل شود. همچنین شما می‌توانید درون خود Azure این عملیات را انجام داده و Git Repository خود را Import نمایید که گاهی این مورد جواب نمی‌دهد.

Azure امکانی که دارد و هر ابزار دیگری این امکان را دارا نمی‌باشد، گزینه Fork می‌باشد. Fork درواقع مثل حالت یک کپی از پروژه جاری در Repository دیگر می‌باشد. به عنوان مثال می‌توانید از یک Git Repo (پروژه Open Source) یک Fork تهیه نمایید و روی یک قسمت آن کار کنید و نتیجه را بدون نیاز به Git Repo و ... به اشتراک بگذارید.

در Azure اگر بخشی از اسم فایل را هم بدانید و سرچ کنید به شما آن فایل را نشان می‌دهد.

همچنین می توان در Azure در فایل README و ... روی یک خط Comment ایجاد نمود. برای این منظور وقتی موس را روی متون حرکت می‌دهید یک علامت + کنار آن ایجاد می‌شود، روی آن کلیک کرده و یک شخص یا Work Item را به آن Assign نمایید.

Repo از نوع SVN را برای استفاده در Azure باید Convert نمایید. پیشنهاد می‌شود که آن را به git تبدیل کنید و Convert را هم انجام ندهید تا در آینده دچار مشکل نشوید. در Azure به طور کلی TFVC و Git وجود دارد .

**»Microsoft Azure DevOps (Session 7) «**

م**قدمه**

در انتهای جلسه ی قبل، ارائه مبحث مربوط به Repoها آغاز شد و در اولین گام ضمن آموزش چگونگی انتقال یک پروژه از Git repository به Repo در Azure، سرفصل های سرویس Repo بررسی اجمالی و موارد File، Commits و Pushes معرفی گردید.



Branches یا انشعاب ها

هر branch موجودیتی است که به یک commit خاص اشاره میکند.

ما از branchها جهت ایجاد یک ساختار Stable استفاده می کنیم. به اینصورت که انشعاب ها به دو دسته:

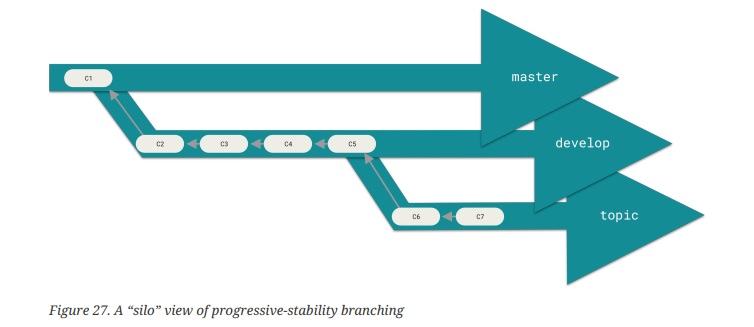
1. Long-Running Branches یا Integration Branch (با طول عمری برابر با طول عمر ریپازیتوری)

و

1. Topic Branches یا Supportive Branches (بصورت مصرفی و با طول عمری برابر با نهایی سازی انشعاب)

تقسیم می شوند.

**نکته:** هیچگاه بر روی دسته Integration Branchها مستقیما کد زده نخواهد شد.



در تصویر فوق از بالا به پایین، هر سطح پایداری و طول عمر بیشتری دارد.

Topic Brachها نقش فرزند را برای Integration Branchها ایجاد می کنند و به محض پایان عملیات توسعه در آن ها و ادغامشان با Branchهای سطح بالا، حذف میگردند.

مطابق با اصول گیت، Branchهای master و develop بصورت Integration و سایر Branchها بصورت Topic تعریف و استفاده می شوند.

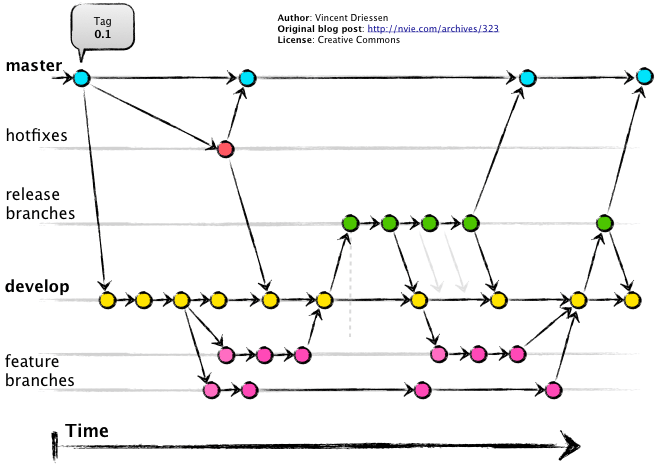


برنچینگ مدل های استاندارد

چندین روش استاندارد برای مدلسازی برنچینگ و ایجاد Back Bone سیستم CI/CD سازمان ها تعریف شده و هر سازمان با توجه به نیاز خود و با در نظر گرفتن سطح تکنولوژی و دانش موجود در آن سازمان، یک روش را مورد استفاده قرار می دهد.

**نکته:** در تمامی مدل ها یکپارچه سازی اصول نامگذاری انشعاب ها در تیم های سازمانی اهمیت دارد.

1. **Git Flow**



این مدل معروف ترین و پر بروکراسی ترین برنچینگ مدل بوده و ساختاری تماما سلسله مراتبی دارد.

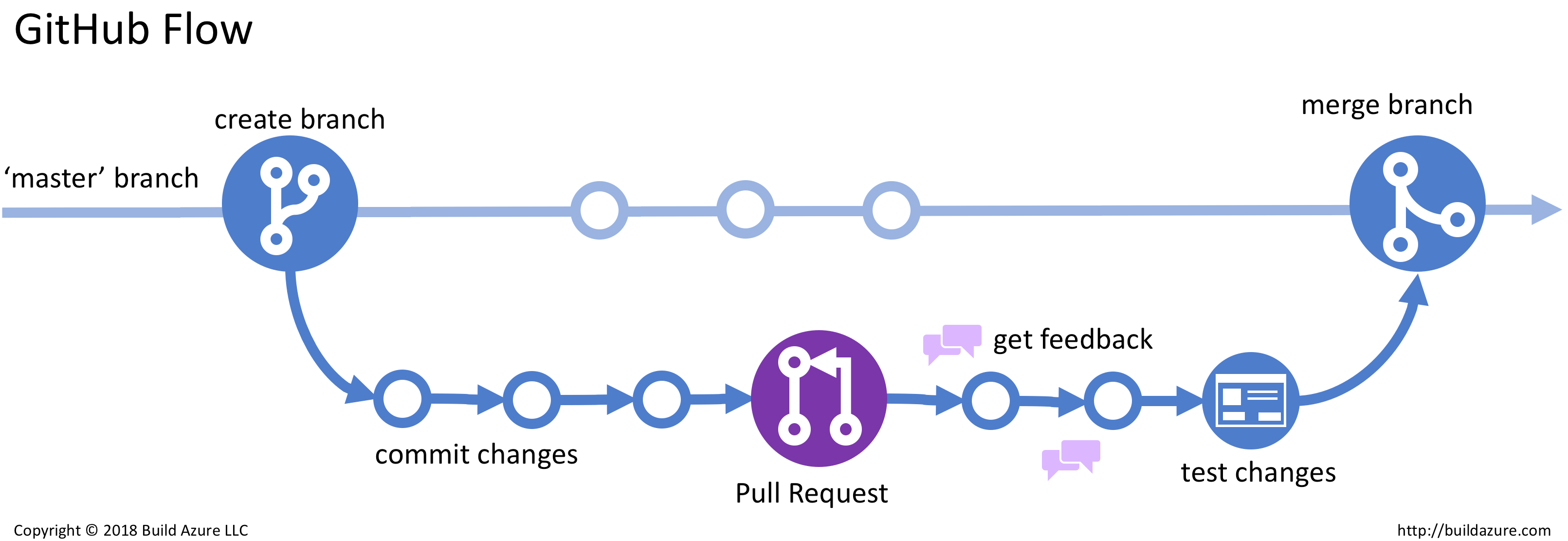
در این مدل مطابق با توضیحات بالا و با هدف ایجاد یک ساختار Stable ما دو دسته انشعاب خواهیم داشت. دسته اول تحت عنوان Integration Branch و دسته دوم تحت عنوان Topic branch می باشد که انشعاب های master و develop از نوع دسته اول و انشعاب های Feature، Release و hotfix از نوع Topic خواهد بود و بلافاصله بعد از ادغام با انشعاب والد حذف خواهند شد.

انشعاب master در این روش برابر با Production سازمان می باشد.  
بنابراین برای کاهش احتمال بروز خطا در سورس اصلی و Production سازمان، نیازمند یک سرپرست با توان فنی و دانش بیشتر جهت مدیریت Pull Request ها می باشیم.

* Pull Request: ابزاری از سرویس Repo می باشد که امکان بررسی انشعاب ها و کدهای مرتبط با آن انشعاب، و همچنین تایید نهایی درخواست Merge از انشعاب سطح پایین تر (فرزند) را بصورت سیستماتیک و دستی فراهم می آورد.

در مدل Git Flow روابط والد و فرزندی بین انشعاب ها و اینکه هر انشعاب با کدامیک از انشعاب های دیگر Merge شود بسیار اهمیت دارد.

1. **Github Flow**

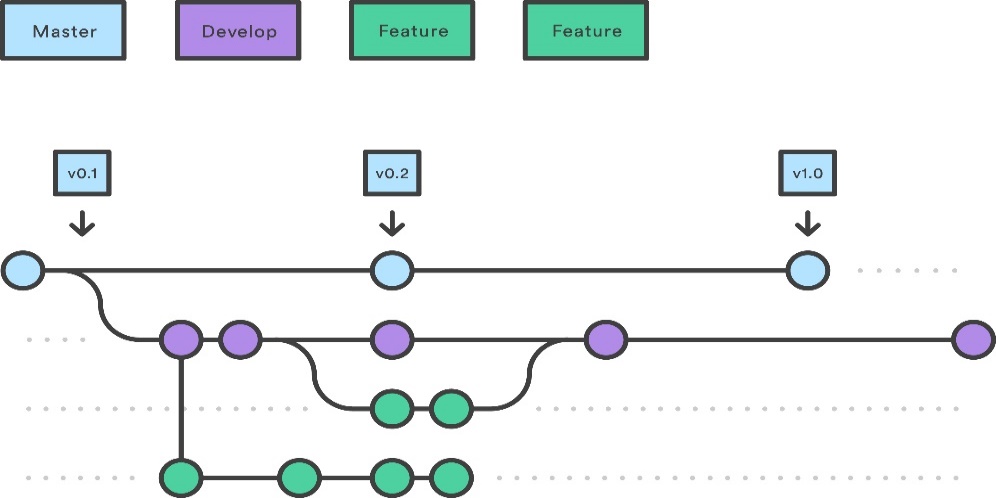


این مدل سبک تر و کم بروکراسی تر از Git Flow می باشد و به اینگونه اجرا می شود که یک انشعاب مرکزی تحت عنوان master خواهیم داشت و مابقی انشعاب ها تحت عنوان Feature Branch در اطراف انشعاب مرکزی ایجاد خواهد شد. هر کدام از انشعاب های Feature پس از تکمیل تغییرات، یک Pull Request ارسال میکنند که این درخواست این امکان را فراهم می آورد که اعضای تیم فیدبک های خود را درباره آن انشعاب مطرح نمایند. پس از دریافت فیدبک ها و اعمال تغییرات و نهایی سازی پروژه، اقدام به Publish کرده و درصورتی که فاز Publish نیز با موفقیت و بدون گزارش خطا طی گردد، کد را در انشعاب مرکزی یا master ادغام (Merge) می نماییم.

باید توجه داشت پروسه Publish مطابق با Pull down time می باشد یعنی لازم است یک زمان استاندارد تعریف گردد و پس از طی شدن آن زمان، در صورت عدم وجود مشکل در ساختار کد، عملیات Merge صورت پذیرد.

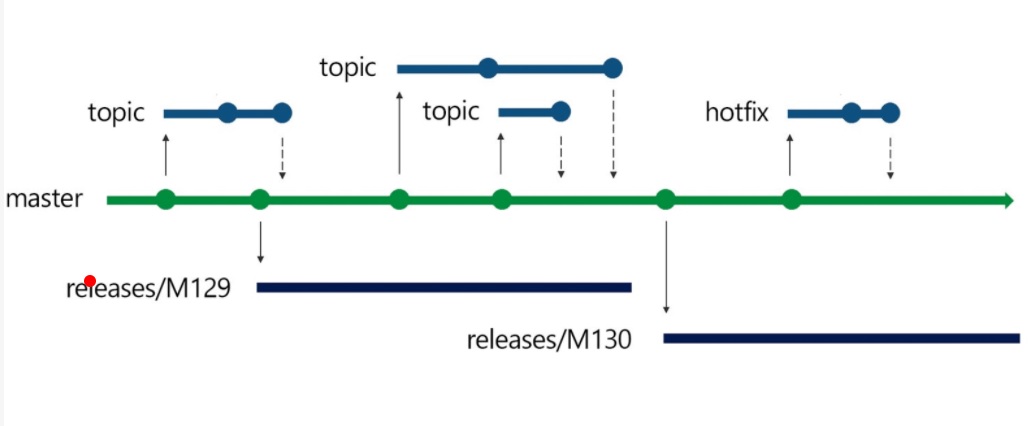
در این مدل بیش از هر چیزی کیفیت سیستم نوبت دهی و فرهنگ رعایت نوبت در سازمان دارای اهمیت می باشد چرا که در صورت عدم رعایت صف انتشار، پس از عملیات Merge احتمال وقوع Conflict و خطاهای اساسی بسیار بالا خواهد بود و سایر انشعاب ها و توسعه دهندگان تحت تاثیر عدم رعایت تقدم و تاخر، قرار خواهند گرفت.

1. **Branch by feature**



مدل Github Flow مطابق با این مدل پیاده سازی شده است.  
به طور کلی مبنای ایجاد انشعاب ها در این مدل، بر اساس Feature Branchها خواهد بود.

1. **Trunk Base**



این مدل برای تیم های با سطح توان فنی و تکنولوژی سطح بالا طراحی شده و مورد استفاده قرار میگیرد.

در این روش مطایق با تصویر، انشعاب های Topic ایجاد و به محض تکمیل بر روی انشعاب master ادغام خواهد شد و پس از آن انشعاب سطح Topic حذف میگردد.

در این روش جهت جلوگیری از ایجاد Conflict و مشکلات سطح کد، از امکان Feature Flag استفاده می گردد.

Feature Flag این امکان را فراهم می سازد که هر انشعابی که بر روی انشعاب master ادغام گردد، مادامی که پرچم آن خاموش است، هیچ تغییری در سورس کد و فایل های موجود در انشعاب مرکزی نخواهیم داشت و بنابراین مشکلی بوجود نخواهد آمد.

در نهایت با روشن کردن Feature Flag مرتبط با یک عملیات ادغام، تاثیرات لحاظ خواهد شد.

در این روش امکان Testing Production فراهم است. یعنی میتوان Feature جدید توسعه داده شده را تنها برای بخشی از کاربران محصول، به نمایش گذاشته و پس از دریافت فیدبک ها و رفع مشکلات احتمالی، بصورت کامل و نهایی عرضه کرد.



شیوه و اصول ایجاد انشعاب ها:

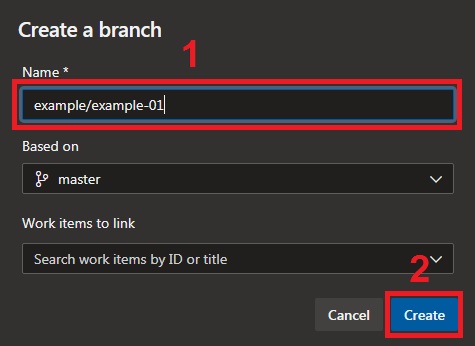
برای ایجاد انشعاب از طریق منوی Branches در سرویس Repo و با کلیک بر روی دکمه آبی رنگ New branch اقدام می نمائیم.

حین ایجاد انشعاب ها باید به فیلد Based on دقت نمائیم. این فیلد جهت تعیین والد برای انشعاب مربوطه مورد استفاده قرار میگیرد.

**نکته:** در مدل Git Flow انشعاب های develop و hotfix بعنوان فرزند انشعاب master تعریف خواهند شد و مابقی انشعاب ها فرزندانی از انشعاب develop خواهند بود.

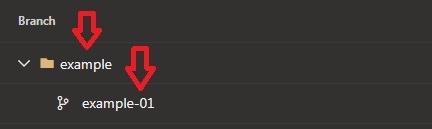
**نکته:** ما فقط انشعاب های master و develop را در root اصلی خواهیم داشت و سایر انشعاب ها می بایست پوشه بندی و نظم دهی شوند.

بنابراین برای دسته بندی انشعاب ها و قرار دهی انشعاب های hotfix، Feature و Release در یک فولدر واحد، هنگام ایجاد یک انشعاب جدید بصورت زیر اقدام می نمائیم:



برای مثال نام انشعاب برابر خواهد بود با example/example-a

در این صورت انشعابی با نام example-a در فولدر example ایجاد خواهد شد. به تصویر زیر دقت نمائید:



**نکته:** برای نمایش همه انشعاب ها در صفحه Branches لازم است از تب های موجود در زیر عنوان صفحه، گزینه All را انتخاب نمائیم. بصورت default تب Mine فعال می باشد که تنها branchهای مرتبط با ما را نمایش می دهد.

در منوی Branches و در ردیف هر انشعاب ستونی تحت نام Behind | Ahead داریم که تعداد commitهایی که آن انشعاب از انشعاب اصلی عقب تر یا جلو تر است را تعیین می نماید. برای جلوگیری از ایجاد تداخل و مشکلات در انشعاب ها می بایست همواره عدد Behind صفر باشد و به نوعی از انشعاب اصلی عقب نباشد.

**نکته:** در صورتی که در فیلد Search brand name نام انشعابی که پاک شده است را جستجو نمائید تا مدت احتمالا 30 روز پس از پاک شدن، امکان Restore نمودن آن انشعاب فراهم است.



Branch policy

برای مدیریت درخواست های Push و Merge بر رو انشعاب های master و develop لازم است از امکان Branch policies استفاده نمائیم.  
برای دسترسی به این صفحه و اعمال تنظیمات باید نشانگر موس خود را بر روی انشعاب master یا develop برده و بر روی آیکون سه نقطه در سمت راست ردیف آن انشعاب، کلیک کنیم و در نهایت با انتخاب گزینه Branch policies وارد صفحه مربوطه می شویم.

**نکته:** با فعالسازی Branch policies امکان انجام عملیات push و Merge را نخواهیم داشت و تنها از طریق Pull Request این امکان فراهم خواهد بود. همچنین تا زمانی که ریپازیتوری حذف نشود، امکان حذف این انشعاب ها میسر نخواهد بود.

**در صفحه Branch policies موارد ذیل جهت اعمال تنظیمات وجود دارد:**

1. **Require a minimum number of reviewers**

با فعال سازی این مورد ضمن تعیین حداقل تعداد Review کنندکان در باکس مربوطه موارد زیر را نیز میتوان تعیین و تنظیم نمود:

1. تعیین اینکه شخصی که درخواست Pull Request را ارسال نموده امکان تایید درخواست خود را داشته باشد یا خیر
2. تعیین اینکه حتی اگر برخی از بازدیدکنندگان رای wait یا reject دادند آیا باز هم میتواند درخواست کامل شود یا خیر
3. و تعیین اینکه آیا بعد از هر commit در انشعاب و اعمال خودکار تغییرات در درخواست Pull، کامنت ها ریست شود یا خیر

**نکته:** زمانی که یه Pull Request ایجاد می گردد، مادامی که آن درخواست به طور کامل تایید یا رد نشده باشد، هر تغییری در انشعاب مربوطه اعمال گردد، در صفحه مربوط به درخواست Pull نیز آن تغییرات را خواهیم داشت و به نوعی Pull Request بصورت خودکار بروزرسانی خواهد شد.

1. **Check for linked work items**

با فعال کردن این گزینه به هر درخواست Pull می بایست حداقل یک Work Item انتشاب داده شود.

1. **Check for comment resolution**

با فعال شدن این گزینه باید همه کامنت ها Resolve شود تا امکان تایید درخواست فراهم گردد

1. **Limit merge types**

با فعال کردن این گزینه می توانیم در نوع عملیات Merge اعمال محدودیت کنیم.

1. **دکمه Add build policy**

میتوانیم یک build جهت بررسی درخواست Pull ایجاد کرده و از این طریق انتساب دهیم.

1. **دکمه Add status policy**

می توان از این طریق ساز و کاری پیاده سازی نمود که بر اساس خروجی یک سرویس دیگر مثل Web API یا سیستم های مانیتورینگ و... مانع اجرای درخواست Pull شد.

1. **دکمه Add automatic reviewers**

از این طریق فرد یا افرادی را تعیین میکنیم که بصورت خودکار در لیست تاییدکنندگان درخواست Pull قرار گیرند.

پس از تعیین و تنظیم موارد فوق بر حسب صلاحدید سازمان، در کنار انشعاب مربوطه در زیر منوی Branches نمادی قرار میگیرد که نشان دهنده وجود policy و عدم امکان Push بر روی آن انشعاب خواهد بود.

C:\Users\m.vakili\Desktop\git\004.png

**نکته:** انشعاب Default (که در تصور فوق قابل مشاهده است) انشعابی است که زمانی که از ریپازیتوری Clone گرفته می شود، بر روی سیستم لوکال بصورت خودکار و به عنوان انشعاب اصلی ایجاد می گردد.



Pull Request یا درخواست ادغام

همانطور که پیش تر اشاره شد Pull Request ابزاری از سرویس Repo می باشد که امکان بررسی انشعاب ها و کدهای مرتبط با آن انشعاب، و همچنین تایید نهایی درخواست Merge از انشعاب سطح پایین تر (فرزند) را بصورت سیستماتیک و دستی فراهم می آورد.

به اینصورت که انشعاب فرزند از طریق سه نقطه موجود در ردیف آن انشعاب و انتخاب گزینه New pull Request می تواند درخواست خود را جهت ادغام به انشعاب مربوطه ارسال نماید. در حین اعمال این درخواست می بایست نام و توضیحات مرتبط با آن و اشخاص Review کننده و ورک آیتم های مرتبط با آن درخواست را وارد و ثبت نمائیم.

پس از ثبت درخواست سرپرست فنی بررسی های لازم را انجام خواهد داد و مطابق با بررسی های خود تایید، اعمال نظر و یا بطور کلی درخواست را رد می نماید.

برای تایید نهایی یک درخواست Pull Request، می بایست تمامی الزامات تعیین شده در Branch policies برقرار شده تا قابلیت ادغام فراهم گردد.

**»Microsoft Azure DevOps (Session 8) «**

ارتباط بین Remote Ripository و Local Repository

زمانی که ما از ریپازیتوری آژور بر روی سیستم لوکال Clone میگیریم، دو انشعاب master و origin master خواهیم داشت.

انشعاب master انشعابی است که ما فعالیت های توسعه ای خود را بر روی آن انجام میدهیم در حالی که انشعاب origin فقط read only بوده و قابل تغییر از سمت لوکال نمی باشد. در نتیجه انشعاب های origin همیشه هستند و به عنوان مرجع مورد استفاده قرار میگیرند.

زمانی که از سیستم لوکال درخواست Pull بر روی یک انشعاب داشته باشیم، تنها انشعاب origin را دریافت میکنیم و نمیتوانیم از آن انشعاب استفاده کنیم مگر آنکه با دستورات Checkout انشعابی از همان انشعاب بر روی لوکال ایجاد و تغییرات مورد نظر را اعمال نمائیم.  
به اینگونه انشعاب ها اصطلاحا Tracking branch میگویند یعنی بین انشعاب لوکال و ریموت اتصال برقرار است و میتوانیم با استفاده از دستورات pull و fetch اطلاعات انشعاب های سمت ریپازیتوری ریموت را دریافت نمائیم.

ما با استفاده از دستور fetch میتوانیم تمامی انشعاب های origin را بروزرسانی کنیم بدون آنکه انشعاب های لوکال تغییری کنند.



Tags

تگ ها در فرآیند های توسعه نرم افزار از طریق Git دارای اهمیت بسیار زیادی می باشند و بر اساس اصول Semantic Versioning پیاده سازی میگردند.

در Semantic Versioning هر ورژن دارای سه بخش مختلف بوده که به صورت زیر با نقطه (دات) از یکدیگر جدا میگردند:

**Major Version . Minor Version . Patches**

عدد Major Version تغییرات اساسی و زیر سیستمی را شامل می شود.

عدد Minor Version تغییراتی مانند افزوده شدن یک Feature را شامل می شوند.

و عدد نهایی (Patches) بیشتر به رفع باگ ها و اختلالات نرم افزار باز می گردند.

جهت مطالعه بیشتر در این خصوص به سند [نسخه بندی معنایی](https://portal.index-holding.com/ict/learnict/%D9%86%D8%B3%D8%AE%D9%87%20%D8%A8%D9%86%D8%AF%DB%8C%20%D9%85%D8%B9%D9%86%D8%A7%DB%8C%DB%8C.docx?d=waeab925ef6e54659ad328f1962389eb3) در آموزش های ICT مراجعه نمائید.

**در فرآیند تگ گذاری (پرچم گذاری) ما 3 استراتژی خواهیم داشت:**

* استراتژی اول براینصورت است که تگ را به یک انشعاب انتساب دهیم. در این روش بصورت خودکار تگ به آخرین commit موجود بر روی آن انشعاب نسبت داده می شود.
* در استراتژی دوم بر اساس یک Commit ID تگ گذاری می کنیم و به یک commit خاص منتسب میکنیم.
* و در استراتژی آخر به یک تگ دیگر انتساب خواهیم داد.

بصورت کلی Tagها هم مانند Branchها به یک commit خاص اشاره میکنند با این تفاوت ها که branchها پاک خواهند شد و Tagها همیشه برای رجوع دوباره ماندگار خواهند ماند، و تفاوت دیگر اینکه Tagها بر خلاف Branchها قابل توسعه نبوده و امکان اعمال تغییرات فراهم نخواهد بود. بنابراین ما در سیستم CI/CD از تگ به منظور بازگردانی نرم افزار و رجوع دوباره به یک پرچم خاص استفاده خواهیم کرد.

ساده ترین شیوه استفاده از Tagها برای مدیریت ورژن، پرچم گذاری بر اساس هر ادغام در انشعاب master می باشد. برای مثال پس از رفع یک باگ و ادغام یک hotfix در master و یا ادغام انشعاب release در master میتوانیم بر اساس حجم و نوع تغییرات ورژن گذاری کنیم.

**نکته:** در فرآیندهای ادغام باید به این نکته توجه داشت که همواره باید نسبت به آخرین تغییرات انشعاب master و develop مطلع بود و ادامه روند توسعه را بر اساس آخرین تغییرات پیش برد. برای مثال پس از رفع یک hotfix و ادغام آن با master لازم است این ادغام با انشعاب های develop و احتمالا feature که در حال توسعه هستند، نیز صورت پذیرد. در غصر اینصورت پس از ادغام feature با master، دوباره باگ قبلی را خواهیم داشت. چراکه انشعاب feature بر روی سورس کد مربوط به قبل از رفع hotfix، توسعه داده شده و تغییرات جدید لحاظ نشده است.



نکات استفاده از ویژوال استودیو برای مدیریت ریپازیتوری

* پس از Clone گرفتن از Repo و از طریق دکمه Branches در ریپازیتوری مربوطه میتوانیم به همه انشعاب های Origin و لوکال دسترسی داشته باشیم. به اینصورت که در ویژوال استودیو لیستی از انشعاب های origin بعلاوه انشعاب اصلی لوکال خواهیم داشت که با دبل کلیک بر روی هر اشنعاب origin، میتوانیم یک انشعاب tracking از آن انشعاب را جهت ویرایش و توسعه کدها بر روی سیستم لوکال داشته باشیم.
* در صورت مواجهه با Conflict در نرم افزار، لیستی از اسناد دارای تداخل را خواهیم داشت که با کلیک بر روی هر کدام دو پنجره تحت عناوین Source و Target خواهیم داشت که با استفاده از چک باکس های کنار هر پنجره میتوانیم صلاحیت هر کدام را تعیین نمائیم.
* در صورتی که کدهای هر دو نسخه از فایل های دارای تداخل صحیح بود و لازم بود هر دو بدون تغییرات، ادغام شوند چک باکس هر دو را فعال میکنیم. به اینصورت که هر کدام از چک باکس ها که زودتر تیک خورد در ردیف های بالاتر کد قرار خواهد گرفت.
* میتوانیم از نوار پایینی نرم افزار با کلیک بر روی نشانگر انشعاب فعال (در سمت راست پنجره) و انتخاب گزینه History تغییرات و commitها را ملاحظه نمائیم.

**نکته:** در فیلد جستجوی Azure این امکان فراهم است با تایپ یک عبارت در تمامی کدها (تمامی ریپازیتوری ها یا یک ریپازیتوری خاص)، تمامی Work Itemها و حتی Wiki به جستجو بپردازیم.

در VS Code افزونه ای به نام Azure Repo داریم که میتوانیم با استفاده از آن محیط ویژوالی شبیه به ویژوال استودیو جهت مدیریت ساده تر و بدون استفاده از دستورات Command بر روی ریپازیتوری های خود داشته باشیم.



امنیت در Repositoryها

جهت دسترسی به تنظیمات امنیتی ریپازیتوری ها از طریق منوی Setting به زیر منوی Repositories در بخش Repo مراجعه می نمائیم.

در این صفحه آپشن ها طی دو طبقه بندی هستند. یک مورد تنظیمات مرتبط با تمامی ریپازیتوری ها و یک مورد تنظیمات مربوط به یک ریپازیتوری خاص.

در تنظیمات Git repositories امکانات امنیتی موجود است که در تنظیمات مرتبط با زیرمجموعه ها قابل دسترسی نیست. برای مثال Create repository که در تنظیمات زیر مجموعه ها در دسترس نیست.

در تنظیمات ریپازیتوری های زیرمجموعه امکانات و محدودیت های مرتبط با همان ریپازیتوری قابل تغییر و تعیین است.

**نکته:** امکان مانیتورینگ افرادی که به ریپازیتوری های مختلف دسترسی دارند فراهم نیست. بنابراین جهت مدیریت بهتر و صحیح تر مباحث امنیتی، لازم است اعضای تیم در گروه های امنیتی معین، تحت نام هایی مشخص، از قبل دسته بندی شده باشند تا هر شخص پس از ایجاد ریپازیتوری، تنها به گروه های مرتبط اعمال دسترسی نماید.



# یکپارچه سازی خط مشی های مرتبط با تمامی ریپازیتوری ها بر اساس یک Branching Model خاص

در منوی Cross-repo policies امکان اعمال خط مشی و محدودیت های مورد نظر، دقیقا مشابه با تنظیمات Branch Policies که در مباحث قبلی مطرح شد، اینبار بصورت کلی و برای تمامی ریپازیتوری ها فراهم است. از این تنظیمات برای یکپارچه سازی برنچینگ مدل سازمان و قرار دادن تمامی تیم ها تحت یک مدل خاص استفاده می شود.

حین ایجاد و اعمال تنظیمات دو آپشن خواهیم داشت که مورد اول تنظیمات را فقط بر روی انشعاب Default برای هر ریپازیتوری اعمال می کند و مورد دوم این امکان را فراهم می آورد تا برای هر انشعاب تحت نام های مورد نظر تنظیماتی جداگانه اعمال نمائیم.

**Microsoft Azure DevOps (Session 10) «**

Pipeline Azure

جهت پیاده سازی و یکپارچه سازی CI/CD (ادغام و تحویل مداوم) و با هدف اجرای خودکار فرآیندهای Test، Build و Delivery سورس کدها – با هر زبان برنامه نویسی – به صورت منظم و پیوسته، مورد استفاده قرار می‌گیرد.



CI (Continuous Integration)

این فرآیند کمک می‌کند تا تغییرات اعمال شده بر روی سورس کد دائما با ریپازیتوری اصلی ادغام گردد. این مهم پس از کسب اطمینان از عدم ناسازگاری مابین کدها یا کامپوننت‌های قدیمی و جدید که توسط تعدادی تست بررسی می‌گردند، نهایی می‌شود و با نام Build pipeline یاد می‌شوند.

**نکته:** همواره فرآیند Build مرتبط به یک ریپازیتوری و یک انشعاب واحد است.



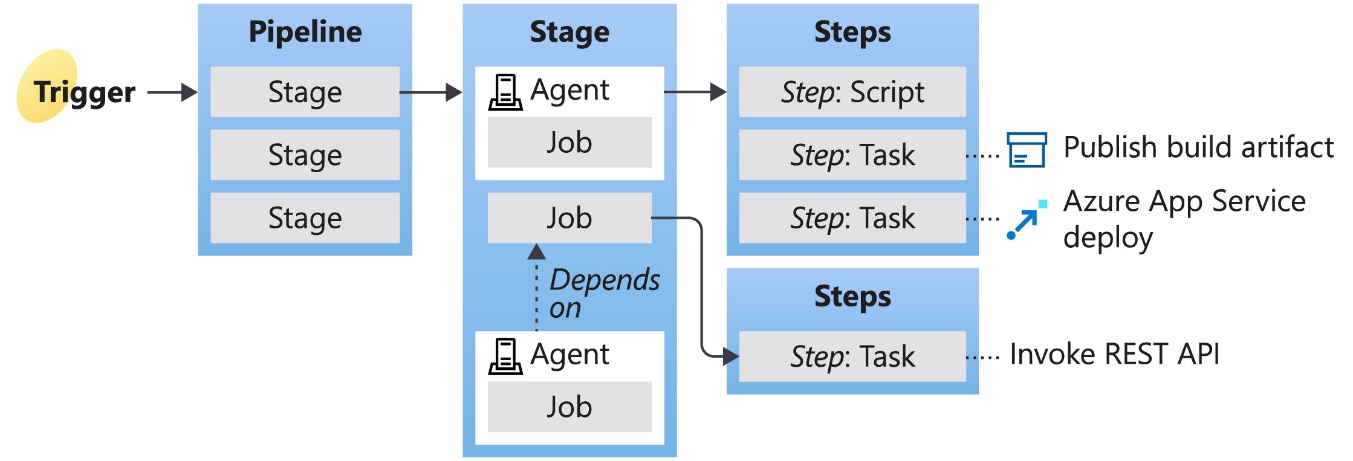
CD (Continuous Delivery (Deployment))

این امکان را به توسعه‌دهندگان می‌دهد تا به صورت روزانه،‌ هفتگی و یا ماهانه نسخهٔ جدید نرم‌افزار خود را بصورت دستی یا خودکار بر روی سرور هدف و در دسترس کاربران قرار دهند و با نام Release pipeline یاد می‌شوند.



پس می‌توان گفت Pipelineها به نوعی Automation مدیریت فعالیت‌های توسعه و انتشار پیوسته نرم افزار می‌باشند.

 کانسپت های کلیدی



 Trigger

از Trigger جهت تنظیم اجرای خودکار Pipeline استفاده می‌شود.

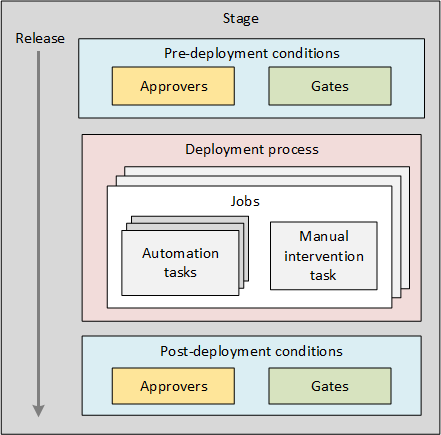


Stage

بعنوان یک مرز منطقی و جداکننده بخش های مختلف فرآیند Release pipeline بوده (مانند Build، QA و Production) و شامل یک یا چند Job می‌باشد. به عبارتی دیگر Stage ارائه دهنده و نماینده محیطی است که قرار است Publish روی آن اتفاق بیفتد.

**نکته:** بصورت دیفالت استیج ها بصورت پشت سر هم اجرا می‌شوند و هر استیج به یک Environment متصل می‌گردد.

**نکته:** برای ورود و خروج از هر Stage میتوان با استفاده از مفاهیم Approver و Gates، محدودیت هایی اعمال کرد. برای مدیریت Approverها بهتر است بجای استفاده از شخص یا اشخاص جهت اعتبار سنجی، از گروه‌های تعریف شده از قبل استفاده گردد. جهت اطلاعات بیشتر [کلیک](https://docs.microsoft.com/en-us/azure/devops/pipelines/release/deploy-using-approvals?view=azure-devops) فرمائید.



******Approval**

مجموعه ای از اعتبار سنجی های مورد نیاز قبل و بعد از فرآیند Deploy سورس کد، بر روی محیط هدف می‌باشد.



Job

هر Stage شمال یک یا چند Job و هر Job شامل یک یا چند Step یا مرحله می‌باشد. هر Job و تمامی Stepهای متعلق به آن بر روی یک Agent اجرا می‌شوند.



Agent

زمانی که Build یا Deployment ما اجرا می‌شود سیستم یک یا چند Job را با کمک Agentها استارت می‌زند. در واقع Agentها نرم افزارهایی هستند که سیستم بوسیله آنها Jobهای مورد نظر خود را اجرا می‌کند. جهت آشنایی بیشتر [کلیک](https://docs.microsoft.com/en-us/azure/devops/pipelines/agents/agents?view=azure-devops&tabs=browser) کنید.

**نکته:** تعامل Agentها با Azure DevOps بصورت رمزنگاری شده می‌باشد.



Pipeline

از یک یا چند Stage تشکیل شده و روند ادغام و استقرار مداوم پروژه را تعریف می‌کند. به بیان دیگر گردش کاری است که نحوه اجرای مراحل Test، Build و Deploy را تعیین کرده و بر روی یک یا چند Environment استقرار می‌یابد.



Environment

هر Environment مجموعه ای از منابع است که بر روی آن فرآیند Deploy بصورت یکپارچه انجام می‌پذیرد و میتواند شامل یک یا چند Virtual Machine یا Kubernetes باشد.

**نکته:** برای اضافه کردن VM به Environmentها مانند Deployment Groupها عمل میکنیم. یعنی بر روی هر VM یک یا چند agent نصب میکنیم تا قابل استفاده باشد.



Deployment group

مجموعه ای از محیط های فیزیکی یا ماشین های هدف جهت Deploy محصول می‌باشد که بر روی هر یک از آن ها Agentهایی نصب شده است. به نوعی یک Deployment group دسته بندی دیگری از Agentها (مانند Agent pool) می‌باشد.

**نکته:** با وجود Pipelineهای نسل جدید و Environmentها این مفهوم کاربرد خود را به نوعی از دست داده است چرا که Deployment groupها فقط در Classic release استفاده می‌شوند.



Step

کوچکترین قسمت و پایین ترین لایه یک Pipeline بوده و میتواند از دو نوع Task و Script باشد.



**Script**

اسکریپت مجموعه ای از کدهای تعریف شده بوده که با استفاده از Command Line، PowerShell یا Bash قادر به اجرای فعالیت های خاص می باشد.



**Task**

هر تسک اسکریپتی پکیج شده می باشد که با استفاده از مجموعه ای از ورودی ها شکل گرفته و عملیاتی میشود.



Run

به هربار اجرای Pipeline گویند که لاگ های Stepها و نتایج Testها را جمع آوری کرده و به محض فراخوانی، ابتدا Pipeline را پردازش کرده و متناسب با آن یک یا چند Agent را مامور به اجرا می کند.



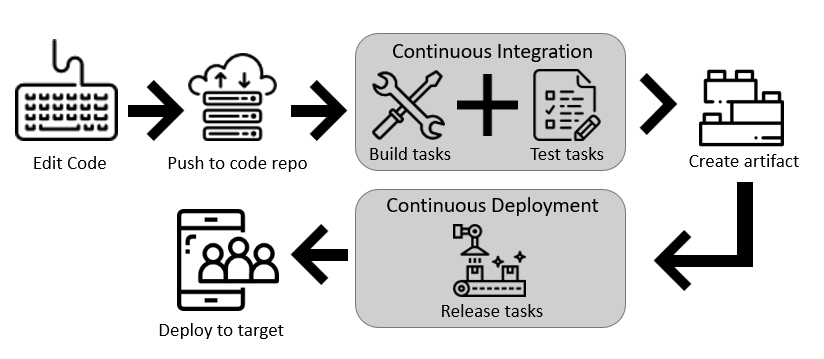
Artifact

خروجی فرآیند CI و به عبارتی نتیجه پروژه تحت مجموعه ای از فایل ها یا پکیج های منتشر شده است که برای اقدامات بعدی مانند توزیع و استقرار (مربوط به CD) مورد استفاده قرار میگیرند.

**برای پیاده سازی فرآیندهای CI/CD با استفاده از آژور دو راه وجود دارد:**

****استفاده از Classic interface

به اینصورت که با استفاده از ادیتور کلاسیک و پس از انتخاب ریپازیتوری و تمپلیت مورد نظر (استفاده از تمپلیت های نمونه یا تمپلیت خام)، یک Build pipeline برای ساخت و تست کدها ایجاد کرده و در انتها بر روی یک Artifact پابلیش میکنیم. از طرف دیگر یک Release pipeline با هدف استقرار آن Artifact بر روی سرورهای هدف ایجاد می نمائیم.

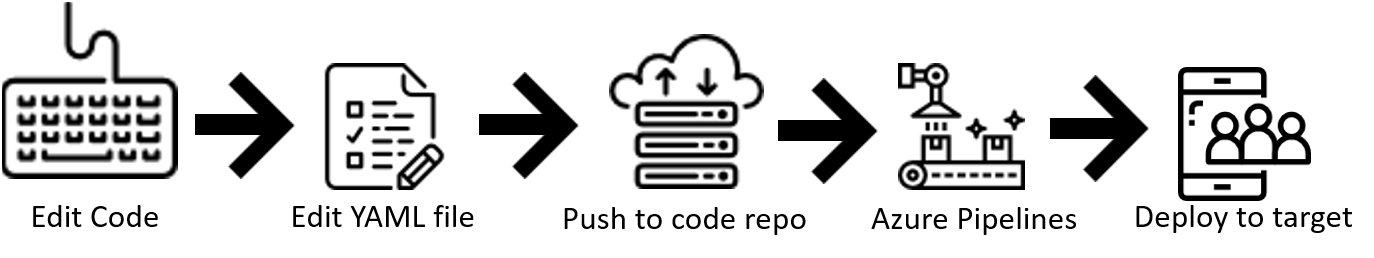




استفاده از YAML syntax

به اینصورت که با استفاده از دستورات و تعاریف موجود در YAML (دستوراتی که مایکروسافت پشتیبانی میکند) فرآیندهای Build، Test و حتی Release خود را پیاده سازی میکنیم.

این دستورات در سندی تحت نام azure-pipelines.yml در ریپازیتوری پروژه نگهداری میشوند.



ذکر این نکته ضروری است که در مدل Classic interface پیاده سازی Build و Release بصورت جداگانه انجام می شود. این در حالیست که در مدل YAML-base میتوان علیرغم پیاده سازی جداگانه این دو فاز، Build و Release را در کنار هم و در یک فایل YAML مدیریت نمود.

* جهت درک بهتر دستورات YAML و آشنایی اولیه با این دستورات به لینک های زیر مراجعه فرمائید:
  + [YAML schema reference](https://docs.microsoft.com/en-us/azure/devops/pipelines/yaml-schema?view=azure-devops&tabs=schema%2Cparameter-schema)
  + [Understand the azure-pipelines.yml file](https://docs.microsoft.com/en-us/azure/devops/pipelines/customize-pipeline?view=azure-devops)



دانلود و نصب Agent جهت اجرای Pipeline

جهت دانلود Agent از مسیر زیر اقدام میکنیم:

Agent > Project Setting > Agent Pools > *Select Agent Pool* > Agents > New Agent > Download

**نکته:** بطور کلی استراتژی مدیریت Agentها و [Agent Pool](https://docs.microsoft.com/en-us/azure/devops/pipelines/agents/pools-queues?toc=%2Fazure%2Fdevops%2Forganizations%2Ftoc.json&bc=%2Fazure%2Fdevops%2Forganizations%2Fbreadcrumb%2Ftoc.json&view=azure-devops&tabs=yaml%2Cbrowser) ها بسیار اهمیت دارد و ایجاد یک Pool به ازای هر پروژه اصلا اقدام درست و حرفه ای نیست. بنابراین با توجه به اینکه در هنگام تعریف Jobها اطلاعات مربوط به Agent pool از ما دریافت میشود، بهتر است در هر Pool چندین Agent داشته باشیم تا علیرغم جلوگیری از ایجاد Poolهای متعدد و در نتیجه مدیریت بهتر Agentها، زمان اجرای Jobها نیز کوتاه تر کرد.

**نکته:** Poolها را در دو سطح تیم پروژه و تیم کالکشن میتوان ایجاد و مدیریت کرد.  
Poolهای سطح کالکشن بصورت جنرال بوده و برای همه تیم پروژه ها به دو صورت قابل استفاده اند. صورت اول اینکه Poolها به محض ایجاد در لیست تیم پروژه ها قابل مشاهده و استفاده اند و در صورت دیگر پس از تعریف، Pool ایجاد شده در لیست تیم پروژه ها نمایش داده نمیشود و باید بصورت دستی و توسط هر تیم پروژه بر حسب صلاحدید تیمی اضافه گردد.

**نکته:** Agentها هنگام اجرا دارای حساسیت بالایی می باشند؛ به همین دلیل حین اجرای build نباید پوشه های مرتبط با آن باز و یا در حال استفاده و اعمال تغییرات (چه از طریق اکسپلور و چه ترمینال) باشند، در غیر اینصورت با خطا مواجه می شویم.

آشنایی با بخش Task groups

Task groupها این امکان را برای ما فراهم می آورند که مجموعه ای از taskهایی که در Build pipelineها و Release pipelineها مورد استفاده قرار داده ایم را گروه بندی کرده و در سایر Pipelineهایمان مورد استفاده مجدد قرار دهیم.

**نکته:** زمانی که ما یک Task group ایجاد کرده ایم و بنابردلایلی به اعمال تغییرات در بخشی از آن گروه میپردازیم، این تغییر بطور خودکار در تمامی Pipeline های حاوی آن Task group اعمال میگردد.

**نکته:** برای ایحاد گروهی از تسک ها لازم است تمامی پارامترها بصورت Variable تعریف شده باشند تا در تمامی موقعیت ها و Pipelineها قابل استفاده باشند.

جهت کسب اطلاعات بیشتر در زمینه task groupها و آشنایی با نحوه ایجاد و مدیریت آنها [کلیک](https://docs.microsoft.com/en-us/azure/devops/pipelines/library/task-groups?view=azure-devops) کنید.



آشنایی با بخش Library

مجموعه ای از دارایی های مرتبط با مراحل Build و Release پروژه می باشد. این دارایی ها شامل [Variable group](https://docs.microsoft.com/en-us/azure/devops/pipelines/library/variable-groups?view=azure-devops&tabs=yaml)ها و [Secure file](https://docs.microsoft.com/en-us/azure/devops/pipelines/library/secure-files?view=azure-devops)ها است.

**»Microsoft Azure DevOps (Session 11) «**



معرفی بخش Pipeline

همانطور که در جلسه پیش اشاره شد دو مدل پیاده سازی Pipeline وجود دارد که هر دو دارای خروجی یکسان بوده و تنها تفاوت در نحوه تعریف و مدیریت Pipeline می باشد.

در روش کلاسیک ما نیاز به تعریف Build pipeline و Release pipeline بصورت جداگانه داریم.

اما در روش جدیدتر یعنی روش YAML base هم میتوان این دو فاز را بصورت جداگانه تعریف کرد (فاز Build با استفاده از YAML و فاز Release با استفاده از متد قدیمی) و هم در قالب یک فایل YAML کامل پیاده سازی نمود.

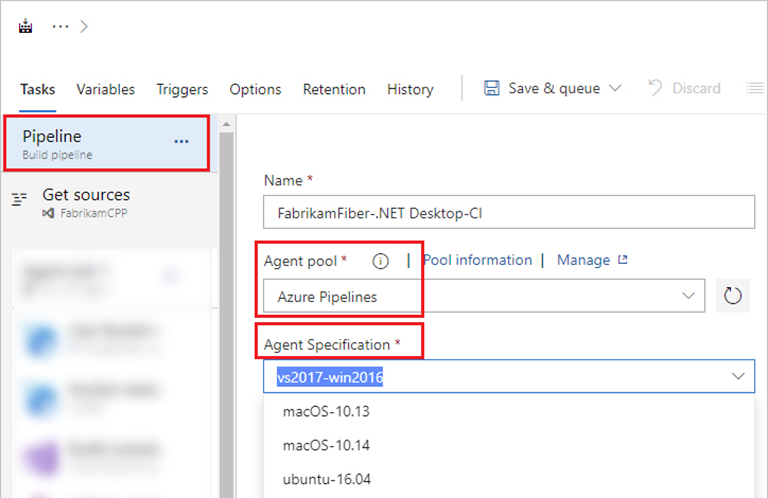
هنگام ایجاد Pipeline و در مرحله پیاده سازی Job، پس از انتخاب ریپازیتوری و تریگر مورد نظر، مایکروسافت تعدادی الگوی از پیش تعریف شده ارائه می دهد که منطبق بر انتخاب ما، چند مورد تسک بصورت خودکار به Job اضافه میکند. در نهایت کاری که باقی می ماند شخصی سازی تسک ها بر اساس نیازمان است.  
اما در صورتی که نخواهیم از این الگوها استفاده کنیم، امکان استفاده از Empty Pipeline نیز برای ما فراهم است.



نکاتی درباره Get resource

* هنگام ایجاد یک Build در تنظیمات Get resource، اگر گزینه Clean را بر روی true قرار دهیم بطور خودکار فایل های موجود در فولدر Build پاکسازی می گردد.
* عدد تعیین شده برای گزینه Shallow fetch (در صورت انتخاب چک باکس) تعداد History که از ریپازیتوری مربوطه برمی گرداند را تعیین میکند. در صورتی که چک باکس آن غیر فعال باشد تمامی History و تمامی انشعاب ها را برمیگرداند. بنابراین هرچه عدد این گزینه کمتر باشد سرعت Build بیشتر خواهد شد.

**نکته :** در مرحله تعریف Jobها برای اجرای Pipeline لازم است Agent pool مورد نظر خود را انتخاب نمائیم.



**نکته:** هر Agent توانایی اجرای تنها یک Job بصورت همزمان را داشته و برای اجرای Multiple Jobها لازم است Agent pool ما حاوی تعداد زیادی Agent نصب شده و فعال باشد.

**زمانی که از طریق Logs به بررسی فرآیند Build میپردازیم به دو دسته Step برخورد میکنیم**

* **دسته اول** مواردی هستند که بصورت عمومی و برای همه Buildها بطور سیستمی و خودکار ایجاد شده و در گام های Jobs که توسط تیم توسعه تعریف و پیاده سازی شده است، وجود ندارند.در این دسته و در قدم اول Build ما Pool و Agent را انتخاب میکند. سپس گام Initialize آغاز شده و ضمن تعیین اسم ماشین و کاربر مورد نظر، شروع به دانلود تسک ها و پکیج هایی که در اختیار ندارد میکند. در گام بعد گیت ریپازیتوری

(به همراه تمامی انشعاب های متعلقه) که توسط ما تعیین شده است را Clone می نماید.

* از این مرحله به بعد، وارد فاز اجرای تسک های تعریف شده و به عبارتی وارد **دسته دوم** Stepها میشویم.

**نکته:** با توجه به مشکلات اتصال به اینترنت شامل سرعت و کیفیت اتصال و همچنین محدودیت های برخواسته از تعریف دسترسی های سازمانی به فضای وب، غالباً مرحله دانلود تسک ها و پکیج های مورد نیاز بسیار زمان بر بوده و در برخی مواقع باعث بروز خطا در فرآیند Build می گردد. برای جلوگیری از این مشکل میتوان پکیج های مورد نیاز را که پیشتر دانلود شده در دایرکتوری ریپازیتوری مورد نظر قرار داد و هنگام تعریف تسک ها، به جای استفاده از امکان دانلود از سایت nuget.org، پکیج ها را بصورت manual جهت استفاده به تسک مربوطه ارائه داد. همچنین میتوان این پکیج ها را با استفاده از Artifactsها کش نمود.

**نکته:** برای Clone کردن ریپازیتوری به ازای هر Build یا Release در دایرکتوری \_work یک پوشه ایجاد می نماید. این اقدام از طرفی سرعت اجرای فرآیند را افزایش میدهد و از طرف دیگر باعث پر شدن فضای هارد و سیستم میشود. به همین دلیل لازم است جهت مدیریت فضای هارد و جلوگیری از تکمیل آن، تسک هایی با هدف پاکسازی فضای پوشه ها و فایل های اضافی تعریف گردند.

**نکته:** همانطور که گفته شد به ازای هر Build یا Release در دایرکتوری \_work یک پوشه با نام گذاری بصورت عددی ایجاد میشود که هر پوشه شامل سه پوشه دیگر تحت نام های a، b و s می باشند که پوشه a حاوی Artifactهاست و فایل های خروجی رو تحت پسوند zip نگهداری میکند. پوشه b مربوط به فایل های باینری مرتبط با بعضی تسک ها می باشد و پوشه s نگهدارنده Source پروژه می باشد.

**نکته:** در صورتی که فرآیند Build با خطا مواجه شود و نهایی نگردد، از طریق صفحه آن Build و بدون مراجعه به جزئیات Jobها میتوان دلیل شکست اجرای Build را مشاهده و در جهت رفع آن اقدام نمود.



**Jobs**

بخش های مختلف جهت کانفیگ و اجرای Pipeline را Jobs گویند. هر Pipeline حداقل داری یک Job می باشد و مجموعه ای از Stepها است که تحت قالب یک واحد و بصورت مکرر و زمانبندی شده اجرا میگردند.

Jobها دارای دو دسته Agent job و Agentless job بوده که دسته اول Jobهایی هستند که با استفاده از یک Agent موجود در یک Agent pool اجرا میگردند و دسته دوم Jobهایی هستند که جهت اجرا نیازی به Agent ندارد و شامل تسک هایی با تعداد کم و محدود می شوند. جهت آشنایی بیشتر با این تسک ها [کلیک](https://docs.microsoft.com/en-us/azure/devops/pipelines/process/phases?view=azure-devops&tabs=classic#agentless-tasks) فرمائید.

**نکته:** میتوان در یک Stage از هر دو دسته Job بصورت همزمان بهره برد. در چنین شرایطی و با هدف ایجاد وابستگی بین این دو نوع Job، مفهوم [Dependencies](https://docs.microsoft.com/en-us/azure/devops/pipelines/process/phases?view=azure-devops&tabs=classic#dependencies) (وابستگی ها) مطرح میگردد.

**نکته:** با استفاده از امکان [Conditions](https://docs.microsoft.com/en-us/azure/devops/pipelines/process/phases?view=azure-devops&tabs=classic#conditions) و [Timeout](https://docs.microsoft.com/en-us/azure/devops/pipelines/process/phases?view=azure-devops&tabs=classic#conditions) نیز میتوان شرایطی جهت اجرای هر Job و فرصت زمانی لازم جهت اجرای آن تعریف و تنظیم نمود. همچنین با استفاده از تنظیمات مربوط به Retention میتوان به مدیریت میزان نگهداشت اطلاعات مربوط به Build پرداخت.

جهت آشنایی با Retention Policy [کلیک](https://docs.microsoft.com/en-us/azure/devops/pipelines/policies/retention?view=azure-devops&tabs=yaml) نمائید.

 Tasks

تسک ها کوچکترین سطح یک Pipeline می باشند. همانطور که در جلسه قبل اشاره شد، هر تسک اسکریپتی پکیج شده بوده که با استفاده از مجموعه ای از ورودی ها شکل گرفته و عملیاتی میشود.

زمانی که ما تسکی را به Pipeline خود اضافه میکنیم در غالب مواقع با مجموعه ای از خواسته ها ([demands](https://docs.microsoft.com/en-us/azure/devops/pipelines/process/demands?view=azure-devops&tabs=yaml)) مواجه می شویم که این خواسته ها به نوعی پیش نیازهای لازم جهت تحویل به Agentها با هدف اجرای (run) آن Task می باشند. برای مثال demands تعیین میکند که یک Build معین فقط از Agentهای قادر به فعالیت بر روی سیستم عامل لینوکس استفاده نماید.  
ازین امکان برای هماهنگی و سازگاری بیشتر بین Buildهایی که از یک Pool یکسان بهره میبرند نیز استفاده میشود و به نوعی ارتباطی بین ماهیت Build و Taskهای آن و Capabilities Agent (توانایی ایجنت ها) برقرار میکند.

**تسک ها بر حسب ماهیت و کارکرد به دسته های زیر تعریف میشوند:**

1. [Build](https://docs.microsoft.com/en-us/azure/devops/pipelines/tasks/?view=azure-devops#build)
2. [Utility](https://docs.microsoft.com/en-us/azure/devops/pipelines/tasks/?view=azure-devops#utility)
3. [Test](https://docs.microsoft.com/en-us/azure/devops/pipelines/tasks/?view=azure-devops#test)
4. [Package](https://docs.microsoft.com/en-us/azure/devops/pipelines/tasks/?view=azure-devops#package)
5. [Deploy](https://docs.microsoft.com/en-us/azure/devops/pipelines/tasks/?view=azure-devops#deploy)
6. [Tool](https://docs.microsoft.com/en-us/azure/devops/pipelines/tasks/?view=azure-devops#tool)

جهت مطالعه در مورد انواع تسک های موجود در هر دسته بندی، بر روی هر کدام کلیک نمائید.

**نکته:** این امکان وجود دارد که با نصب Extensionهای مختلف بر روی آژور، از تسک هایی خارج از تسک های از پیش تعریف شده در دسته بندی های بالا بهره برد.

**نکته:** Task groupها با توجه به دسته بندی انتخابی در هنگام تعریف، در یکی از دسته بندی های بالا قرار گرفته و قابل انتخاب و استفاده اند اما امکان تغییر جزئیات آنها فقط از طریق بخش Task groups فراهم است.

لازم به ذکر است، میتوان به جای استفاده از تسک های تعریف شده، از Scriptها جهت تعیین فعالیت های مورد نظر استفاده برد.

**نکته:** Step پایانی هر Job معمولاً یک تسک ویژه تحت عنوان Publish Artifact می باشد که هم میتواند Artifact تولید شده را در خود آژور نگهداری نموده و هم با دریافت یک دایرکتوری و نام معین از ما (دایرکتوری و نام میتوانند متغیر باشند)، آرتیفکت تولید شده را تحت آن نام و در دایرکتوری تعیین شده جایگذاری می نماید.  
جهت ساده سازی رجوع دوباره به Artifactها و تفکیک دقیق آنها برای استفاده های آتی، بهتر است از اسم و شماره Build در نام گذاری هایمان استفاده نمائیم.  
این تسک به نوعی پل بین فرآیندهای CI و CD ما می باشد.

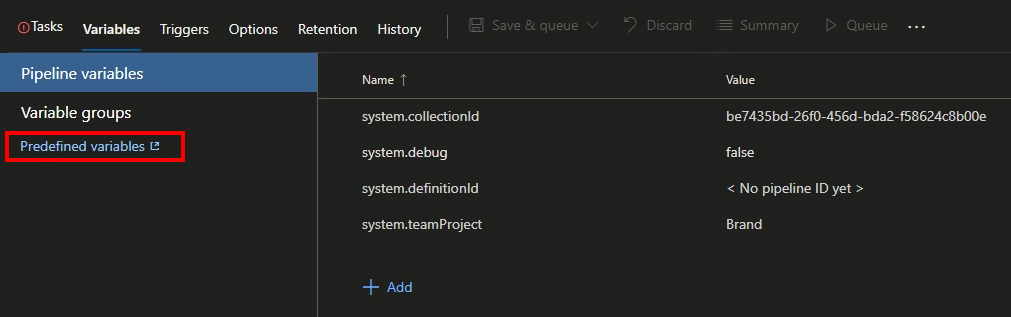
**نکته:** نگهداری Artifactها در آژور از طرفی امن تر بوده و نیاز به مطرح شدن Network share دیگر نیست. اما از طرفی دیگر باعث پر شدن و سنگین شدن دیتابیس ما می شود.



Variables

متغیرها برای دریافت، تغییر و انتقال مقادیر کلیدی در بین گام های مختلف یک Pipeline و به نوعی برای مدیریت بهتر پارامترها مورد استفاده قرار گرفته و بصورت رشته ای (String) ذخیره میشوند.

متغیرها دارای دو دسته Predefined و Custom بوده که متغیرهای نوع Predefined از طریق تب Variables در هنگام ایجاد Jobها قابل دسترسی می باشند.



* همچنین از طریق لینک های زیر نیز میتوانید به متغیرهای این دسته بندی دسترسی داشته باشید.
  + [predefined variables](https://docs.microsoft.com/en-us/azure/devops/pipelines/build/variables?view=azure-devops&tabs=yaml)
  + [Classic release and artifacts variables](https://docs.microsoft.com/en-us/azure/devops/pipelines/release/variables?view=azure-devops&tabs=batch)



History

از این قسمت جهت مانیتور کردن تغییرات که توسط چه کسی و چه زمانی اتفاق افتاده است، استفاده میکنیم. با استفاده از گزینه ی Compare Difference به مقایسه مراحل و با استفاده از Revert Pipeline به مرحله مورد نظر باز میگردیم.

اما نوع دوم پیاده سازی Pipelineها استفاده از دستورات YAML می باشد.  
هنگام استفاده از متد YAML، فایل آن بر روی روت ریپازیتوری Push شده و در همانجا نگهداری میشود. از این رو میتوان بدون نیاز به استفاده از Azure و با استفاده از ابزارهایی نظیر VSCode، Visual Studio و... نیز اقدام به اعمال تغییرات بر روی آن نمود. Azure بطور خودکار تغییرات را تریگر کرده و اعمال میکند.  
در این روش این امکان فراهم است که هر دو فاز Build و Release در یک فایل YAML پیاده سازی و مدیریت شوند. در این روش با اعمال تغییرات و ذخیره آن، Run بصورت خودکار اتفاق می افتد.



سلسله مراتب پیاده سازی یک فایل YAML

* Pipeline
  + Stage A
    - Job 1
      * Step 1.1
      * Step 1.2
      * ...
    - Job 2
      * Step 2.1
      * Step 2.2
      * ...
  + Stage B
    - ...

**نکته:** هر بلاک از دستورات YAML توسط دکمه Setting (در بالای آن بلاک) بصورت ویژوال مدیریت می شود که به نوعی ابزاری جهت ساده سازی فرآیند کدنویسی بر اساس YAML و تغییر برخی مقادیر می باشد.

**نکته:** Task groupها در متد YAML کاربردی نداشته و فقط برای مدل Classic مورد استفاده قرار میگیرند.



Status Badge

نشان دهنده موفقیت یا شکست آخرین اقدام اجرایی فرآیندهای CI یا CD بوده که غالباً برای بکارگیری در Wikiها مورد استفاده قرار میگیرند.

C:\Users\DFA.CO\AppData\Local\Microsoft\Windows\INetCache\Content.Word\build-status-badges.png

**»Microsoft Azure DevOps (Session12) «**



معرفی بخش Environment

هر [Environment](https://docs.microsoft.com/en-us/azure/devops/pipelines/process/environments?view=azure-devops) مجموعه ای از منابع است که هرکدام به عنوان هدف استقرار یک Pipeline بوده و بر روی آنها فرآیند Deploy بصورت یکپارچه انجام می‌پذیرد.

این مفهوم دارای دو مسئولیت مهم می‌باشند:

1. محیط هایی که قرار است جهت اجرای فرآیند deploy مورد استفاده قرار گیرند را به تیم توسعه ارائه می‌دهد.
2. امکاناتی جهت مدیریت کانسپت Approverها در اختیار تیم توسعه می‌گذارند.

در حال حاضر دو نوع Environment قابل تعریف می‌باشد:

1. [Kubernetes resource](https://docs.microsoft.com/en-us/azure/devops/pipelines/process/environments-kubernetes?view=azure-devops)
2. [Virtual machine resource](https://docs.microsoft.com/en-us/azure/devops/pipelines/process/environments-virtual-machines?view=azure-devops)

**نکته:** این امکان فراهم است تا یک Environment خالی ایجاد کنیم و بعداً به آن منابع را اختصاص دهیم.

**نکته:** Environmentها تنها برای پایپلاین‌های YAML base مورد استفاده قرار می‌گیرند.

**نکته:** می‌توان با استفاده از تگ‌ها جهت دسته بندی کامپیوترها و منابع یک Environment اقدام نمود.  
مثلا جدا از اینکه با استفاده از خود Environmentها میتوان این فیلتر را ایجاد کرد که کدام ماشین ها کدام Pipelineها را اجرا کنند، با استفاده از تگ‌ها هم می‌توان فیلتر دیگری افزود و تعیین کرد که در یک Environment مشخص فقط ماشین‌های دارای یک تگ خاص، Pipeline را اجرا نمایند



Deployment job

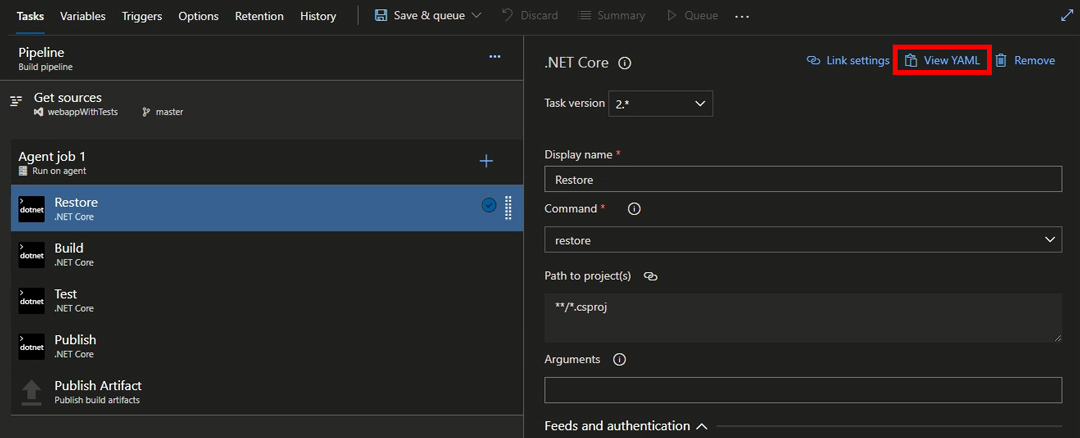
در مدل YAML Pipeline توصیه می‌شوند که گام‌های deploy خود را در نوعی ویژه از job تحت عنوان deployment job قرار دهید. یک deployment job مجموعه‌ای از Stepها است که به طور متوالی در برابر Environmentها اجرا می‌شوند.



تبدیل Classic pipeline به YAML

برای این تبدیل امکان ویژه‌ای فراهم است که هر Task قابلیت تحویل فایل YAML مرتبط با خود را دارد (مطابق با تصویر زیر).

بنابراین برای تکمیل این تبدیل باید دستورات YAML تمامی تسک ها را یک به یک دریافت کرده و به ترتیب در فایل YAML جایگذاری نمائیم. مواردی از نوع Script نیز عیناً به فایل YAML منتقل می‌شوند.



**نکته:** برای تبدیل Task groupها به دستورات YAML می‌بایست وارد آن شده و تمامی Taskها را بصورت جداگانه تبدیل نمائیم.



استفاده از Template به جای Task group در YAML

در متد YAML این امکان فراهم است تا از دستورات خود یک الگو ایجاد نموده و به جای Task groupها که در متد Classic مورد استفاده قرار می‌گرفتند، استفاده نمائیم.  
بنابراین YAMLها یک mode پیچیده تری دارند که می‌شود templateهای گوناگون پیاده سازی کرده و هر زمان که لازم به استفاده بود آنها را فراخوانی نمود.



**تنظیمات Agent Pool**

Jobهایی که توسط آن Agent pool انجام شده است را لیست می‌کند.

**Agents**

تعداد Agentهای نصب شده بر روی pool و وضعیت هرکدام را نشان می‌دهد.

Online: یعنی agent آماده به اجرا است.

Offline: یعنی agent قابل استفاده نیست.

Enabled: در صورت غیرفعال بودن امکان استفاده از agent فراهم نیست حتی اگر online باشد.

**Security**

جهت مدیریت تخصیص و استفاده از Agentها بکار می‌رود.

در دو سطح کالکشن و تیم پروژه قابل تعریف است که هر سطح دارای سه Role جداگانه می‌باشد:

در سطح کاکشن:

1. Reader

این نقش تنها امکان مانیتورینگ Agentها و بررسی صحت کارکرد آنها را فراهم می‌آورد.

1. Service Account

این نقش این امکان را فراهم می‌آورد تا اشخاص با استفاده از Agent poolهای سطح کالکشن یا سازمان، Agent poolهایی در سطح پروژه ایجاد نمایند.

1. این سطح دسترسی کامل جهت ایجاد و حذف Agentها از Pool مورد نظر را داشته و می‌توانند نقش سایر اشخاص را در این سطح تعیین و تنظیم نمایند. شخصی که Pool را ایجاد می‌کند بطور معمول در این نقش قرار می‌گیرد.

در سطح تیم پروژه:

1. Reader

امکان مشاهده Agent pool را داشته و بطور معمول جهت اضافه نمودن اشخاصی با هدف مانیتورینگ build و deployment jobs بکار می‌روند.

1. User

این نقش دسترسی استفاده از Agentها را در زمان ایجاد Pipelineها دارند.

1. Administrator

این نقش امکان مدیریت تماس اشخاص و roleهای تخصیصی به آنها را دارد. شخصی که Pool را ایجاد می‌کند بطور معمول در این نقش قرار می‌گیرد.



Maintenance job

مفهوم [Maintenance job](https://docs.microsoft.com/en-us/azure/devops/pipelines/agents/pools-queues?view=azure-devops&tabs=yaml%2Cbrowser#what-is-a-maintenance-job) در Agent pool که از تب setting قابل تعریف و شخصی سازی است، جهت پاکسازی دایرکتوری ها و ریپازیتوری های قدیمی بصورت دوره ای بکار می‌رود. این اقدام جهت مدیریت و افزایش فضای خالی دیسک انجام می‌پذیرد.  
Maintenance job در سطح کالکشن تنظیم می‌گردد.

Maintenance job timeout: حداکثر زمانی که یک Agent می‌تواند پیوسته به فعالیت Maintenance بپردازد.

Maintenance Percentage: درصدی از Agentها که بصورت همزمان می‌توانند فعالیت Maintenance را انجام دهند.



Maintenance history

از تب Maintenance history نیز می‌توان این فعالیت را مانیتور کرده و لاگ آن را دانلود و بررسی نمود.



Secure fileها در بخش library

شامل فایل‌هایی می‌شوند که تیم توسعه و یا سازمان تمایلی به نگهداری آن‌ها بر روی ریپازیتوری خود نداشته اما در فرآیند Release باید مورد استفاده قرار گیرند. مانند Certificates، Android Keystore files و SSH keys

بکارگیری این فایل ها در فرآیند Release با استفاده از Task مرتبط با دانلود Secure fileها صورت می‌پذیرد.

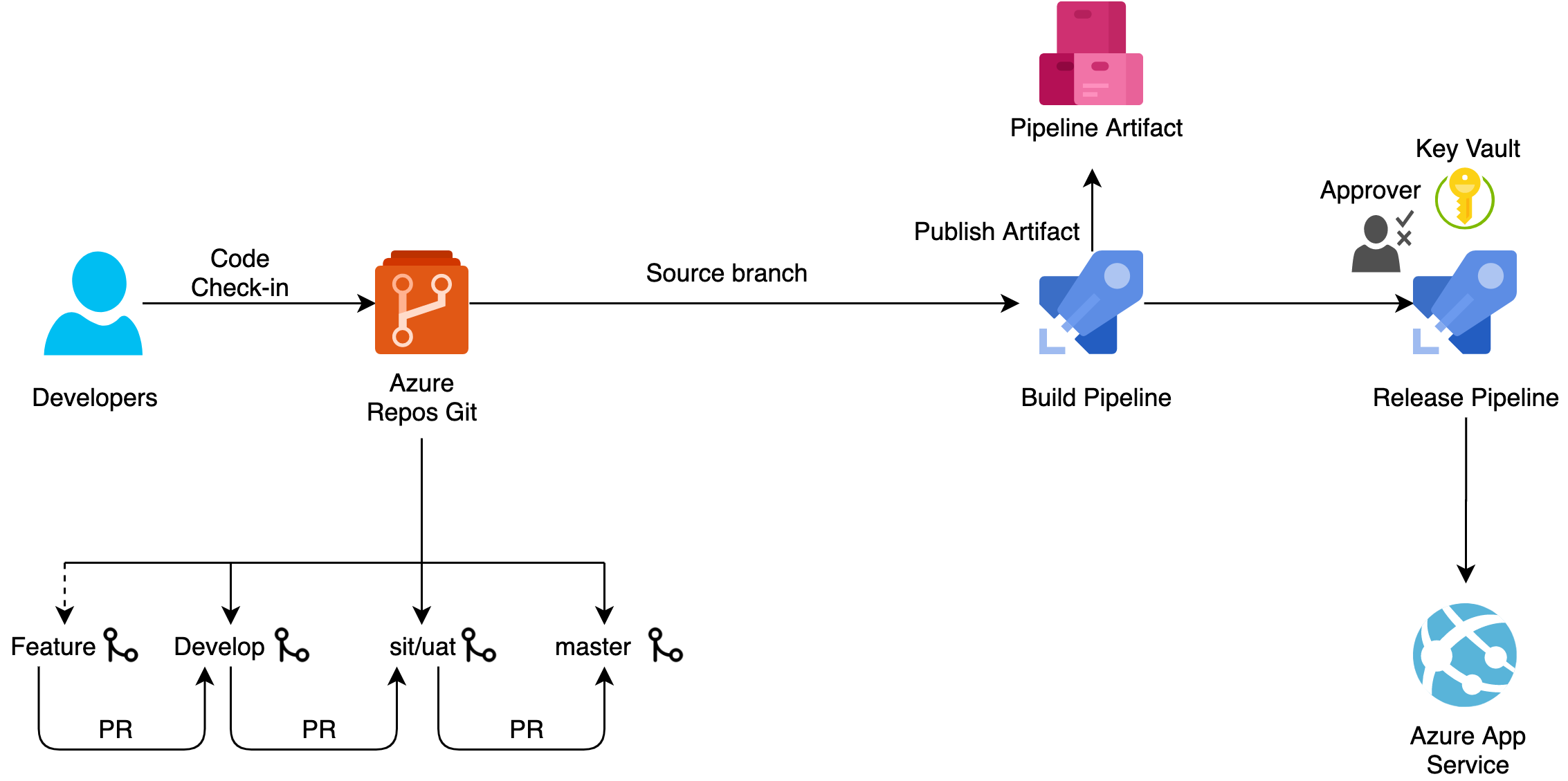
**نکته:** هنگامی که Secure file به library اضافه می‌گردد، محتوای آن قابل رویت و ویرایش نیست و صرفا جهت فراخوانی در Release بکار می‌روند.

مطالعه بیشتر از طریق [این لینک](https://docs.microsoft.com/en-us/azure/devops/pipelines/library/secure-files?view=azure-devops#:~:text=Secure%20files%20are%20defined%20and,referencing%20them%20from%20a%20task.)

**«Microsoft Azure Devops (session 13)»**

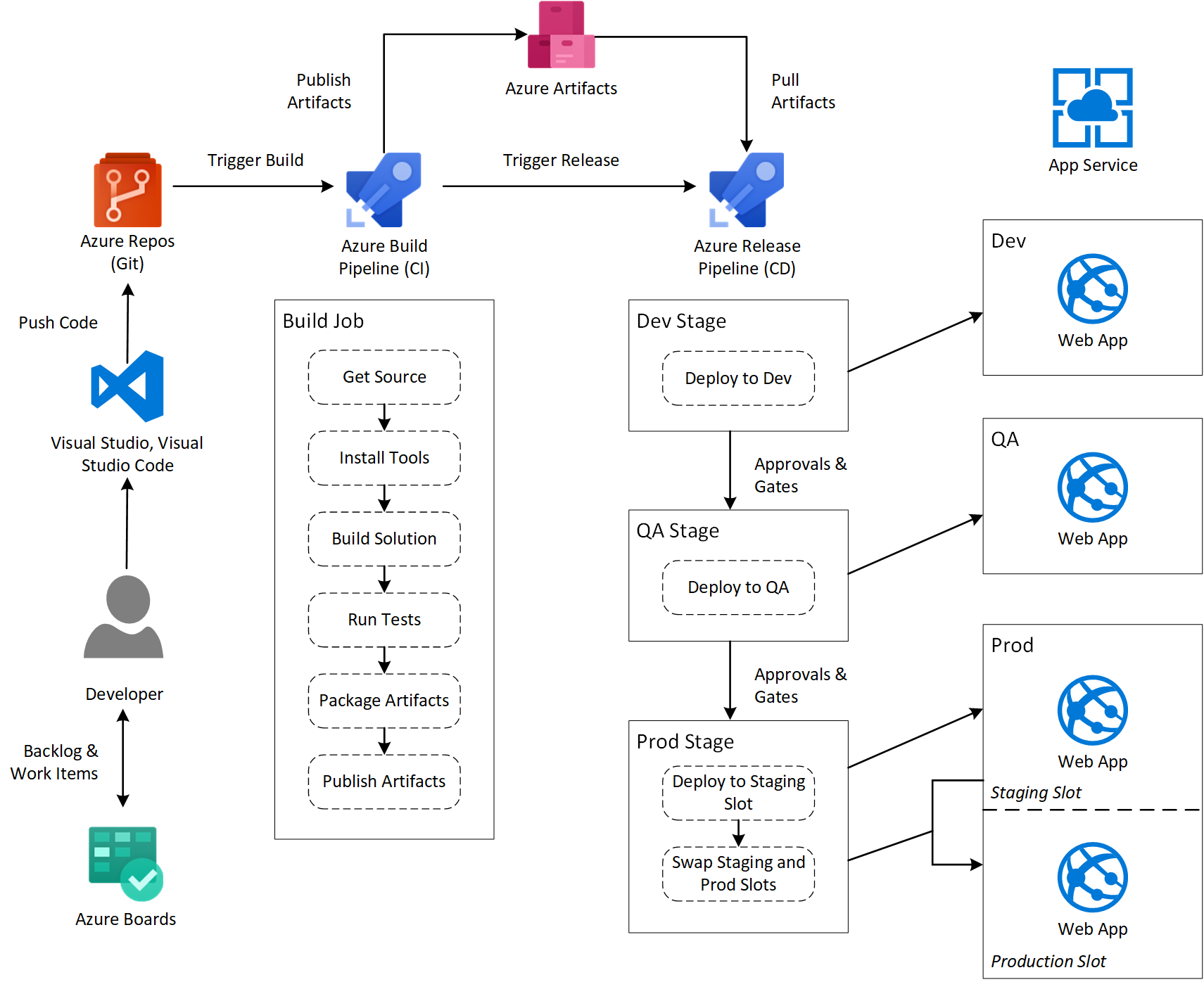


شرح CICD در آژور



* Developer سورس کد خود را به ریپازیتوری منتقل می‌کند.
* پروژه دارای چند انشعاب مختلف بوده که توسط Pull request انتقلات مابینشان اتفاق می‌افتد.
* می‌توان این انشعاب‌ها را برای اجرای فرآیند CI تریگر کرد.
* با اعمال تغییر در انشعاب تریگر شده، فرآیند build اجرا می‌گردد و در صورت نیاز Artifact آن تولید و نگهداری می‌شود.
* در این مرحله خروجی فرآیند Build را به عنوان ورودی به فرآیند Release منتقل می‌کنیم.
* با استفاده از Approverها فرآیند Release را مدیریت می‌نمائیم.
* در نهایت پروژه را به سرور مورد نظر انتقال می‌دهیم.

اکنون مورد دیگری را بررسی نمائیم:

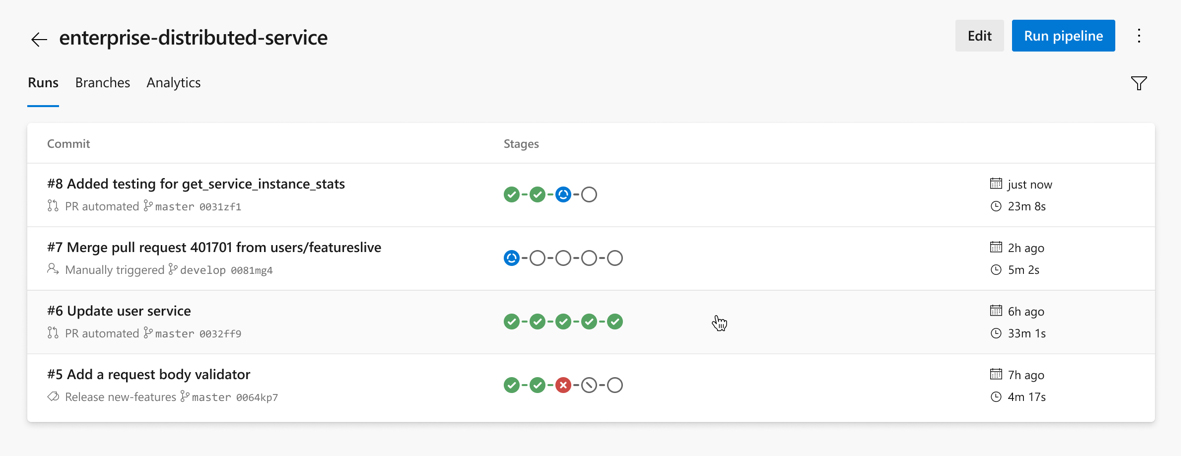


در این مورد:

* در سرویس Board آژور کارهای پروژه تعریف می‌گردد.
* منطبق بر کارها، توسعه دهنده در محیط توسعه خود اقدام به نهایی سازی اقدامات می‌نماید.
* پروژه بر روی ریپازیتوری مرتبط Push می‌شود.
* فرآیند Build را بر روی انشعاب هدف تریگر می‌کنیم.
* با اعمال تغییرات بر روی انشعاب، بطور خودکار Build اتفاق می‌افتد.
* حین اجرای CI، نیازمندی‌های فرآیند دانلود و نصب می‌گردد.
* Taskهای تعریف شده و Testها اجرا می‌گردند.
* Artifact آن ایجاد د در نهایت Publish می‌شود.
* در این مرحله Artifact جهت اجرای فرآیند Release به فاز CD منتقل و این فاز تریگر می‌شود.
* Taskهای موجود در Stageها به ترتیب و با پشت سر گذاشتن Approverها و Gateها اجرا می‌گردند.
* در صورت اخذ تمامی تاییدات در آخرین گام، پروژه به سرور یا سرورهای مورد نظر انتقال می‌یابد.



وضعیت های Build در تب Runs



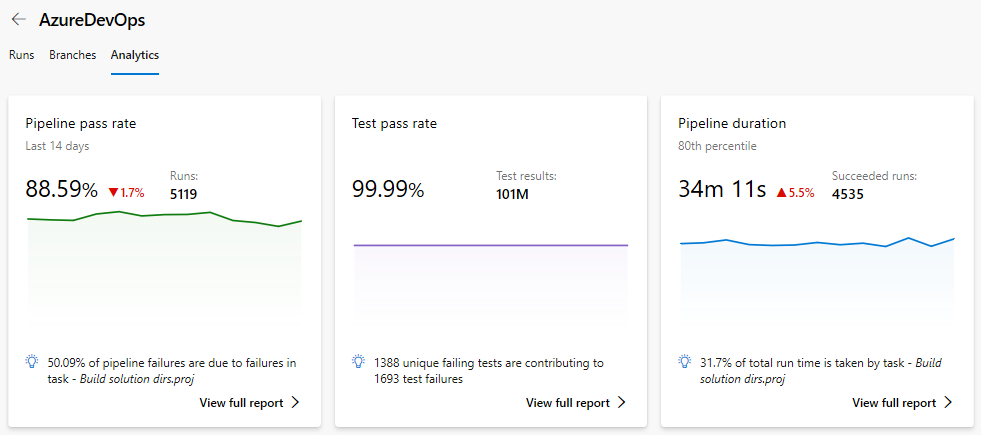
* علامت سفید Buildهای انجام نشده (skip شده) را نمایش می‌دهد.
* علامت سبز نشان دهنده Buildهای موفق می‌باشد.
* علامت آبی نشان دهنده Buildهای در حال اجرا می‌باشد.
* علامت قرمز هم به Buildهای Fail شده اشاره دارد.

**«Microsoft Azure Devops (session 14)»**



گزارشات در Pipelineها

با استفاده از تب Analytics در Pipeline می‌توان گزارشات تجمیعی از میزان پیشرفت و موفقیت آن بدست آورد.



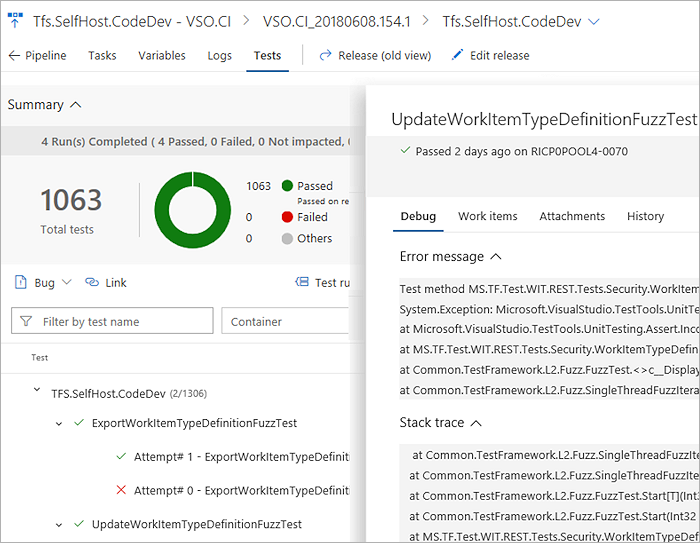
1. بلوک اول تحت عنوان Pipeline pass rate درصد موفقیت آمیز بودن Buildهای Pipeline را نمایش داده و تعیین می‌کند که کدام یک از Taskها باعث ایجاد شکست در فرآیند Build شده که از این اطلاعات برای بهبود و جایگزینی Taskها میتوان بهره برد.
2. بلوک دوم تحت عنوان Test pass rate درصد موفقیت آمیز بودن Testهای Pipeline را نمایش داده و نمایی از Testهای با بیشترین شکست را در اختیار کاربر می‌گذارد.
3. بلوک سوم با نام Pipeline duration نشان دهنده زمان صرف شده برای اجرای کامل و موفق Build بوده و Stepهایی که زمان زیادی از اجرای فرآیند را به خود اختصاص داده‌اند، به ترتیب نمایش می‌دهد که این اطلاعات برای بهبود اینگونه task‌ها بسیار کاربردی می باشد.

**نکته:** با استفاده از فیلترهای موجود در صفحه میتوان بازه زمانی ارائه گزارش را تغییر داد.

**نکته:** آپشن Analytics برای تک تک Pipelineها و Buildهای ما موجود و قابل استفاده است.

جهت مطالعه بیشتر در این زمینه [کلیک](https://docs.microsoft.com/en-us/azure/devops/pipelines/reports/pipelinereport?view=azure-devops) نمائید.

همچنین با استفاده از تب Tests در Pipeline، گزارشی کامل از روند اجرای تست‌ها و میزانی از موفقیت و شکست آنها دریافت می‌کنیم که با کلیک بر روی نام تست، گزارشات عمیق‌تر شده و آمار جزئی تری به کاربر ارائه می‌دهد.



این موضوع در این [لینک](https://docs.microsoft.com/en-us/azure/devops/pipelines/test/review-continuous-test-results-after-build?view=azure-devops) به تفصیل بیان شده است.



Retention Policy

همانطور که در جلسه 11 بطور خلاصه اشاره شد، Retention Policyها تعیین می‌کنند که به چه میزان testها، releaseها، runها و... در سیستم و دیتابیس نگهداری شوند و به نوعی برای مدیریت فضای سیستم کاربرد دارند.

این مفهوم در سه سطح زیر کاربرد دارد:سثحش

1. Pipeline

تعیین می‌کند چه زمانی Artifactها، Attachmentها، runها و Pull requestها نگهداری شود.

1. Release (Classic)

برای تنظیم زمان نگهداشت Buildها بکار می‌روند.

1. Test

تعیین کننده زمان نگهداشت تست‌های خودکار و دستی، نتایج مرتبط با آنها و Attachmentهاست.

جهت مطالعه بیشتر در این زمینه [کلیک](https://docs.microsoft.com/en-us/azure/devops/pipelines/policies/retention?view=azure-devops&tabs=yaml) نمائید.