Microsoft

Microsoft Official Course

AZ-400T06

Implementing Continuous Feedback

AZ-400T06

Implementing Continuous Feedback

Information in this document, including URL and other Internet Web site references, is subject to change without notice. Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

The names of manufacturers, products, or URLs are provided for informational purposes only and Microsoft makes no representations and warranties, either expressed, implied, or statutory, regarding these manufacturers or the use of the products with any Microsoft technologies. The inclusion of a manufacturer or product does not imply endorsement of Microsoft of the manufacturer or product. Links may be provided to third party sites. Such sites are not under the control of Microsoft and Microsoft is not responsible for the contents of any linked site or any link contained in a linked site, or any changes or updates to such sites. Microsoft is not responsible for webcasting or any other form of transmission received from any linked site. Microsoft is providing these links to you only as a convenience, and the inclusion of any link does not imply endorsement of Microsoft of the site or the products contained therein.

MICROSOFT LICENSE TERMS
MICROSOFT INSTRUCTOR-LED COURSEWARE

These license terms are an agreement between Microsoft Corporation (or based on where you live, one of its affiliates) and you. Please read them. They apply to your use of the content accompanying this agreement which includes the media on which you received it, if any. These license terms also apply to Trainer Content and any updates and supplements for the Licensed Content unless other terms accompany those items. If so, those terms apply.

BY ACCESSING, DOWNLOADING OR USING THE LICENSED CONTENT, YOU ACCEPT THESE TERMS. IF YOU DO NOT ACCEPT THEM, DO NOT ACCESS, DOWNLOAD OR USE THE LICENSED CONTENT.

If you comply with these license terms, you have the rights below for each license you acquire.

1. **DEFINITIONS.**
   a) "Authorized Learning Center" means a Microsoft IT Academy Program Member, Microsoft Learning Competency Member, or such other entity as Microsoft may designate from time to time.
   b) "Authorized Training Session" means the instructor-led training class using Microsoft Instructor-Led Courseware conducted by a Trainer at or through an Authorized Learning Center.
   c) "Classroom Device" means one (1) dedicated, secure computer that an Authorized Learning Center owns or controls that is located at an Authorized Learning Center's training facilities that meets or exceeds the hardware level specified for the particular Microsoft Instructor-Led Courseware.
   d) "End User" means an individual who is (i) duly enrolled in and attending an Authorized Training Session or Private Training Session, (ii) an employee of a MPN Member, or (iii) a Microsoft full-time employee.
   e) "Licensed Content" means the content accompanying this agreement which may include the Microsoft Instructor-Led Courseware or Trainer Content.
   f) "Microsoft Certified Trainer" or "MCT" means an individual who is (i) engaged to teach a training session to End Users on behalf of an Authorized Learning Center or MPN Member, and (ii) currently certified as a Microsoft Certified Trainer under the Microsoft Certification Program.
   g) "Microsoft Instructor-Led Courseware" means the Microsoft-branded instructor-led training course that educates IT professionals and developers on Microsoft technologies. A Microsoft Instructor-Led Courseware title may be branded as MOC, Microsoft Dynamics or Microsoft Business Group courseware.
   h) "Microsoft IT Academy Program Member" means an active member of the Microsoft IT Academy Program.
   i) "Microsoft Learning Competency Member" means an active member of the Microsoft Partner Network program in good standing that currently holds the Learning Competency status.
   j) "MOC" means the "Official Microsoft Learning Product" instructor-led courseware known as Microsoft Official Course that educates IT professionals and developers on Microsoft technologies.
   k) "MPN Member" means an active Microsoft Partner Network program member in good standing.
   l) "Personal Device" means one (1) personal computer, device, workstation or other digital electronic device that you personally own or control that meets or exceeds the hardware level specified for the particular Microsoft Instructor-Led Courseware.

m) "Private Training Session" means the instructor-led training classes provided by MPN Members for corporate customers to teach a predefined learning objective using Microsoft Instructor-Led Courseware. These classes are not advertised or promoted to the general public and class attendance is restricted to individuals employed by or contracted by the corporate customer.

n) Trainer" means (i) an academically accredited educator engaged by a Microsoft IT Academy Program Member to teach an Authorized Training Session, and/or (ii) a MCT.

o) "Trainer Content" means the trainer version of the Microsoft Instructor-Led Courseware and additional supplemental content designated solely for Trainers' use to teach a training session using the Microsoft Instructor-Led Courseware. Trainer Content may include Microsoft PowerPoint presentations, trainer preparation guide, train the trainer materials, Microsoft One Note packs, classroom setup guide and Pre-release course feedback form. To clarify, Trainer Content does not include any software, virtual hard disks or virtual machines.

2. **USE RIGHTS.** The Licensed Content is licensed not sold. The Licensed Content is licensed on a ***one copy per user*** basis, such that you must acquire a license for each individual that accesses or uses the Licensed Content.

2.1. Below are five separate sets of use rights. Only one set of rights apply to you.

a) **If you are a Microsoft IT Academy Program Member:**

i) Each license acquired on behalf of yourself may only be used to review one (1) copy of the Microsoft Instructor-Led Courseware in the form provided to you. If the Microsoft Instructor-Led Courseware is in digital format, you may install one (1) copy on up to three (3) Personal Devices. You may not install the Microsoft Instructor-Led Courseware on a device you do not own or control.

ii) For each license you acquire on behalf of an End User or Trainer, you may either:

1. distribute one (1) hard copy version of the Microsoft Instructor-Led Courseware to one (1) End User who is enrolled in the Authorized Training Session, and only immediately prior to the commencement of the Authorized Training Session that is the subject matter of the Microsoft Instructor-Led Courseware being provided, **or**

2. provide one (1) End User with the unique redemption code and instructions on how they can access one (1) digital version of the Microsoft Instructor-Led Courseware, **or**

3. provide one (1) Trainer with the unique redemption code and instructions on how they can access one (1) Trainer Content,

provided you comply with the following:

iii) you will only provide access to the Licensed Content to those individuals who have acquired a valid license to the Licensed Content,

iv) you will ensure each End User attending an Authorized Training Session has their own valid licensed copy of the Microsoft Instructor-Led Courseware that is the subject of the Authorized Training Session,

v) you will ensure that each End User provided with the hard-copy version of the Microsoft Instructor-Led Courseware will be presented with a copy of this agreement and each End User will agree that their use of the Microsoft Instructor-Led Courseware will be subject to the terms in this agreement prior to providing them with the Microsoft Instructor-Led Courseware. Each individual will be required to denote their acceptance of this agreement in a manner that is enforceable under local law prior to their accessing the Microsoft Instructor-Led Courseware,

vi) you will ensure that each Trainer teaching an Authorized Training Session has their own valid licensed copy of the Trainer Content that is the subject of the Authorized Training Session,

vii) you will only use qualified Trainers who have in-depth knowledge of and experience with the Microsoft technology that is the subject of the Microsoft Instructor-Led Courseware being taught for all your Authorized Training Sessions,

viii) you will only deliver a maximum of 15 hours of training per week for each Authorized Training Session that uses a MOC title, and

ix) you acknowledge that Trainers that are not MCTs will not have access to all of the trainer resources for the Microsoft Instructor-Led Courseware.

b) **If you are a Microsoft Learning Competency Member:**

i) Each license acquired on behalf of yourself may only be used to review one (1) copy of the Microsoft Instructor-Led Courseware in the form provided to you. If the Microsoft Instructor-Led Courseware is in digital format, you may install one (1) copy on up to three (3) Personal Devices. You may not install the Microsoft Instructor-Led Courseware on a device you do not own or control.

ii) For each license you acquire on behalf of an End User or MCT, you may either:

1. distribute one (1) hard copy version of the Microsoft Instructor-Led Courseware to one (1) End User attending the Authorized Training Session and only immediately prior to the commencement of the Authorized Training Session that is the subject matter of the Microsoft Instructor-Led Courseware provided, **or**
2. provide one (1) End User attending the Authorized Training Session with the unique redemption code and instructions on how they can access one (1) digital version of the Microsoft Instructor-Led Courseware, **or**
3. you will provide one (1) MCT with the unique redemption code and instructions on how they can access one (1) Trainer Content,

provided you comply with the following:

iii) you will only provide access to the Licensed Content to those individuals who have acquired a valid license to the Licensed Content,

iv) you will ensure that each End User attending an Authorized Training Session has their own valid licensed copy of the Microsoft Instructor-Led Courseware that is the subject of the Authorized Training Session,

v) you will ensure that each End User provided with a hard-copy version of the Microsoft Instructor-Led Courseware will be presented with a copy of this agreement and each End User will agree that their use of the Microsoft Instructor-Led Courseware will be subject to the terms in this agreement prior to providing them with the Microsoft Instructor-Led Courseware. Each individual will be required to denote their acceptance of this agreement in a manner that is enforceable under local law prior to their accessing the Microsoft Instructor-Led Courseware,

vi) you will ensure that each MCT teaching an Authorized Training Session has their own valid licensed copy of the Trainer Content that is the subject of the Authorized Training Session,

vii) you will only use qualified MCTs who also hold the applicable Microsoft Certification credential that is the subject of the MOC title being taught for all your Authorized Training Sessions using MOC,

viii) you will only provide access to the Microsoft Instructor-Led Courseware to End Users, and

ix) you will only provide access to the Trainer Content to MCTs.

c) **If you are a MPN Member:**

i) Each license acquired on behalf of yourself may only be used to review one (1) copy of the Microsoft Instructor-Led Courseware in the form provided to you. If the Microsoft Instructor-Led Courseware is in digital format, you may install one (1) copy on up to three (3) Personal Devices. You may not install the Microsoft Instructor-Led Courseware on a device you do not own or control.

ii) For each license you acquire on behalf of an End User or Trainer, you may either:

1. distribute one (1) hard copy version of the Microsoft Instructor-Led Courseware to one (1) End User attending the Private Training Session, and only immediately prior to the commencement of the Private Training Session that is the subject matter of the Microsoft Instructor-Led Courseware being provided, **or**
2. provide one (1) End User who is attending the Private Training Session with the unique redemption code and instructions on how they can access one (1) digital version of the Microsoft Instructor-Led Courseware, **or**
3. you will provide one (1) Trainer who is teaching the Private Training Session with the unique redemption code and instructions on how they can access one (1) Trainer Content,

provided you comply with the following:

iii) you will only provide access to the Licensed Content to those individuals who have acquired a valid license to the Licensed Content,

iv) you will ensure that each End User attending an Private Training Session has their own valid licensed copy of the Microsoft Instructor-Led Courseware that is the subject of the Private Training Session,

v) you will ensure that each End User provided with a hard copy version of the Microsoft Instructor-Led Courseware will be presented with a copy of this agreement and each End User will agree that their use of the Microsoft Instructor-Led Courseware will be subject to the terms in this agreement prior to providing them with the Microsoft Instructor-Led Courseware. Each individual will be required to denote their acceptance of this agreement in a manner that is enforceable under local law prior to their accessing the Microsoft Instructor-Led Courseware,

vi) you will ensure that each Trainer teaching an Private Training Session has their own valid licensed copy of the Trainer Content that is the subject of the Private Training Session,

vii) you will only use qualified Trainers who hold the applicable Microsoft Certification credential that is the subject of the Microsoft Instructor-Led Courseware being taught for all your Private Training Sessions,

viii) you will only use qualified MCTs who hold the applicable Microsoft Certification credential that is the subject of the MOC title being taught for all your Private Training Sessions using MOC,

ix) you will only provide access to the Microsoft Instructor-Led Courseware to End Users, and

x) you will only provide access to the Trainer Content to Trainers.

d) **If you are an End User:**

For each license you acquire, you may use the Microsoft Instructor-Led Courseware solely for your personal training use. If the Microsoft Instructor-Led Courseware is in digital format, you may access the Microsoft Instructor-Led Courseware online using the unique redemption code provided to you by the training provider and install and use one (1) copy of the Microsoft Instructor-Led Courseware on up to three (3) Personal Devices. You may also print one (1) copy of the Microsoft

Instructor-Led Courseware. You may not install the Microsoft Instructor-Led Courseware on a device you do not own or control.

e) **If you are a Trainer.**

i) For each license you acquire, you may install and use one (1) copy of the Trainer Content in the form provided to you on one (1) Personal Device solely to prepare and deliver an Authorized Training Session or Private Training Session, and install one (1) additional copy on another Personal Device as a backup copy, which may be used only to reinstall the Trainer Content. You may not install or use a copy of the Trainer Content on a device you do not own or control. You may also print one (1) copy of the Trainer Content solely to prepare for and deliver an Authorized Training Session or Private Training Session.

ii) You may customize the written portions of the Trainer Content that are logically associated with instruction of a training session in accordance with the most recent version of the MCT agreement. If you elect to exercise the foregoing rights, you agree to comply with the following: (i) customizations may only be used for teaching Authorized Training Sessions and Private Training Sessions, and (ii) all customizations will comply with this agreement. For clarity, any use of "*customize*" refers only to changing the order of slides and content, and/or not using all the slides or content, it does not mean changing or modifying any slide or content.

2.2. **Separation of Components.** The Licensed Content is licensed as a single unit and you may not separate their components and install them on different devices.

2.3. **Redistribution of Licensed Content.** Except as expressly provided in the use rights above, you may not distribute any Licensed Content or any portion thereof (including any permitted modifications) to any third parties without the express written permission of Microsoft.

2.4. **Third Party Notices.** The Licensed Content may include third party code that Microsoft, not the third party, licenses to you under this agreement. Notices, if any, for the third party code are included for your information only.

2.5. **Additional Terms.** Some Licensed Content may contain components with additional terms, conditions, and licenses regarding its use. Any non-conflicting terms in those conditions and licenses also apply to your use of that respective component and supplements the terms described in this agreement.

3. **LICENSED CONTENT BASED ON PRE-RELEASE TECHNOLOGY.** If the Licensed Content's subject matter is based on a pre-release version of Microsoft technology ("**Pre-release**"), then in addition to the other provisions in this agreement, these terms also apply:

a) **Pre-Release Licensed Content.** This Licensed Content subject matter is on the Pre-release version of the Microsoft technology. The technology may not work the way a final version of the technology will and we may change the technology for the final version. We also may not release a final version. Licensed Content based on the final version of the technology may not contain the same information as the Licensed Content based on the Pre-release version. Microsoft is under no obligation to provide you with any further content, including any Licensed Content based on the final version of the technology.

b) **Feedback.** If you agree to give feedback about the Licensed Content to Microsoft, either directly or through its third party designee, you give to Microsoft without charge, the right to use, share and commercialize your feedback in any way and for any purpose. You also give to third parties, without charge, any patent rights needed for their products, technologies and services to use or interface with any specific parts of a Microsoft technology, Microsoft product, or service that includes the feedback. You will not give feedback that is subject to a license that requires Microsoft to license its technology, technologies, or products to third parties because we include your feedback in them. These rights survive this agreement.

c) **Pre-release Term.** If you are an Microsoft IT Academy Program Member, Microsoft Learning Competency Member, MPN Member or Trainer, you will cease using all copies of the Licensed Content on the Pre-release technology upon (i) the date which Microsoft informs you is the end date for using the Licensed Content on the Pre-release technology, or (ii) sixty (60) days after the commercial release of the technology that is the subject of the Licensed Content, whichever is earliest ("**Pre-release term**"). Upon expiration or termination of the Pre-release term, you will irretrievably delete and destroy all copies of the Licensed Content in your possession or under your control.

4. **SCOPE OF LICENSE.** The Licensed Content is licensed, not sold. This agreement only gives you some rights to use the Licensed Content. Microsoft reserves all other rights. Unless applicable law gives you more rights despite this limitation, you may use the Licensed Content only as expressly permitted in this agreement. In doing so, you must comply with any technical limitations in the Licensed Content that only allows you to use it in certain ways. Except as expressly permitted in this agreement, you may not:

   - access or allow any individual to access the Licensed Content if they have not acquired a valid license for the Licensed Content,
   - alter, remove or obscure any copyright or other protective notices (including watermarks), branding or identifications contained in the Licensed Content,
   - modify or create a derivative work of any Licensed Content,
   - publicly display, or make the Licensed Content available for others to access or use,
   - copy, print, install, sell, publish, transmit, lend, adapt, reuse, link to or post, make available or distribute the Licensed Content to any third party,
   - work around any technical limitations in the Licensed Content, or
   - reverse engineer, decompile, remove or otherwise thwart any protections or disassemble the Licensed Content except and only to the extent that applicable law expressly permits, despite this limitation.

5. **RESERVATION OF RIGHTS AND OWNERSHIP.** Microsoft reserves all rights not expressly granted to you in this agreement. The Licensed Content is protected by copyright and other intellectual property laws and treaties. Microsoft or its suppliers own the title, copyright, and other intellectual property rights in the Licensed Content.

6. **EXPORT RESTRICTIONS.** The Licensed Content is subject to United States export laws and regulations. You must comply with all domestic and international export laws and regulations that apply to the Licensed Content. These laws include restrictions on destinations, end users and end use. For additional information, see www.microsoft.com/exporting.

7. **SUPPORT SERVICES.** Because the Licensed Content is "as is", we may not provide support services for it.

8. **TERMINATION.** Without prejudice to any other rights, Microsoft may terminate this agreement if you fail to comply with the terms and conditions of this agreement. Upon termination of this agreement for any reason, you will immediately stop all use of and delete and destroy all copies of the Licensed Content in your possession or under your control.

9. **LINKS TO THIRD PARTY SITES.** You may link to third party sites through the use of the Licensed Content. The third party sites are not under the control of Microsoft, and Microsoft is not responsible for the contents of any third party sites, any links contained in third party sites, or any changes or updates to third party sites. Microsoft is not responsible for webcasting or any other form of transmission received from any third party sites. Microsoft is providing these links to third party sites to you

only as a convenience, and the inclusion of any link does not imply an endorsement by Microsoft of the third party site.

10. **ENTIRE AGREEMENT.** This agreement, and any additional terms for the Trainer Content, updates and supplements are the entire agreement for the Licensed Content, updates and supplements.

11. **APPLICABLE LAW.**

    a) United States. If you acquired the Licensed Content in the United States, Washington state law governs the interpretation of this agreement and applies to claims for breach of it, regardless of conflict of laws principles. The laws of the state where you live govern all other claims, including claims under state consumer protection laws, unfair competition laws, and in tort.

    b) Outside the United States. If you acquired the Licensed Content in any other country, the laws of that country apply.

12. **LEGAL EFFECT.** This agreement describes certain legal rights. You may have other rights under the laws of your country. You may also have rights with respect to the party from whom you acquired the Licensed Content. This agreement does not change your rights under the laws of your country if the laws of your country do not permit it to do so.

13. **DISCLAIMER OF WARRANTY. THE LICENSED CONTENT IS LICENSED "AS-IS" AND "AS AVAILABLE." YOU BEAR THE RISK OF USING IT. MICROSOFT AND ITS RESPECTIVE AFFILIATES GIVES NO EXPRESS WARRANTIES, GUARANTEES, OR CONDITIONS. YOU MAY HAVE ADDITIONAL CONSUMER RIGHTS UNDER YOUR LOCAL LAWS WHICH THIS AGREEMENT CANNOT CHANGE. TO THE EXTENT PERMITTED UNDER YOUR LOCAL LAWS, MICROSOFT AND ITS RESPECTIVE AFFILIATES EXCLUDES ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT.**

14. **LIMITATION ON AND EXCLUSION OF REMEDIES AND DAMAGES. YOU CAN RECOVER FROM MICROSOFT, ITS RESPECTIVE AFFILIATES AND ITS SUPPLIERS ONLY DIRECT DAMAGES UP TO US$5.00. YOU CANNOT RECOVER ANY OTHER DAMAGES, INCLUDING CONSEQUENTIAL, LOST PROFITS, SPECIAL, INDIRECT OR INCIDENTAL DAMAGES.**

    This limitation applies to

    - anything related to the Licensed Content, services, content (including code) on third party Internet sites or third-party programs; and
    - claims for breach of contract, breach of warranty, guarantee or condition, strict liability, negligence, or other tort to the extent permitted by applicable law.

    It also applies even if Microsoft knew or should have known about the possibility of the damages. The above limitation or exclusion may not apply to you because your country may not allow the exclusion or limitation of incidental, consequential or other damages.

Please note: As this Licensed Content is distributed in Quebec, Canada, some of the clauses in this agreement are provided below in French.

Remarque : Ce le contenu sous licence étant distribué au Québec, Canada, certaines des clauses dans ce contrat sont fournies ci-dessous en français.

EXONÉRATION DE GARANTIE. Le contenu sous licence visé par une licence est offert « tel quel ». Toute utilisation de ce contenu sous licence est à votre seule risque et péril. Microsoft n'accorde aucune autre garantie expresse. Vous pouvez bénéficier de droits additionnels en vertu du droit local sur la protection dues consommateurs, que ce contrat ne peut modifier. La ou elles sont permises par le droit locale, les garanties implicites de qualité marchande, d'adéquation à un usage particulier et d'absence de contrefaçon sont exclues.

LIMITATION DES DOMMAGES-INTÉRÊTS ET EXCLUSION DE RESPONSABILITÉ POUR LES DOMMAGES. Vous pouvez obtenir de Microsoft et de ses fournisseurs une indemnisation en cas de dommages directs uniquement à hauteur de 5,00 $ US. Vous ne pouvez prétendre à aucune indemnisation pour les autres dommages, y compris les dommages spéciaux, indirects ou accessoires et pertes de bénéfices.

Cette limitation concerne:

- tout ce qui est relié au le contenu sous licence, aux services ou au contenu (y compris le code) figurant sur des sites Internet tiers ou dans des programmes tiers; et.
- les réclamations au titre de violation de contrat ou de garantie, ou au titre de responsabilité stricte, de négligence ou d'une autre faute dans la limite autorisée par la loi en vigueur.

Elle s'applique également, même si Microsoft connaissait ou devrait connaître l'éventualité d'un tel dommage. Si votre pays n'autorise pas l'exclusion ou la limitation de responsabilité pour les dommages indirects, accessoires ou de quelque nature que ce soit, il se peut que la limitation ou l'exclusion ci-dessus ne s'appliquera pas à votre égard.

EFFET JURIDIQUE. Le présent contrat décrit certains droits juridiques. Vous pourriez avoir d'autres droits prévus par les lois de votre pays. Le présent contrat ne modifie pas les droits que vous confèrent les lois de votre pays si celles-ci ne le permettent pas.

Revised November 2014

Contents

Module 0 Welcome

Start Here

Azure DevOps Curriculum

Welcome to the **Implementing Continuous Feedback** course. This course is part of a series of courses to help you prepare for the AZ-400, **Microsoft Azure DevOps Solutions**[1] certification exam.

The DevOps certification exam is for DevOps professionals who combine people, process, and technologies to continuously deliver valuable products and services that meet end user needs and business objectives. DevOps professionals streamline delivery by optimizing practices, improving communications and collaboration, and creating automation. They design and implement strategies for application code and infrastructure that allow for continuous integration, continuous testing, continuous delivery, and continuous monitoring and feedback.

Exam candidates must be proficient with Agile practices. They must be familiar with both Azure administration and Azure development and experts in at least one of these areas. Azure DevOps professionals must be able to design and implement DevOps practices for version control, compliance, infrastructure as code, configuration management, build, release, and testing by using Azure technologies.

AZ-400 Study Areas	Weights
Implement DevOps Development Processes	20-25%
Implement Continuous Integration	10-15%
Implement Continuous Delivery	10-15%
Implement Dependency Management	5 -10%
Implement Application Infrastructure	15-20%
Implement Continuous Feedback	10-15%
Design a DevOps Strategy	20-25%

There are seven exam study areas. Each study area has a corresponding course. While it is not required that you have completed any of the other courses in the DevOps series before taking this course, it is highly recommended that you start with the first course in the series, and progress through the courses

1 https://www.microsoft.com/en-us/learning/exam-AZ-400.aspx

in order. In particular, make sure you have completed the Design a DevOps Strategy course so that you get a big-picture view of DevOps before drilling down into specific methodologies.

✓ This course will focus on preparing you for the **Implement Continuous Feedback** area of the AZ-400 certification exam.

About this Course

Course Description

This course provides the knowledge and skills to implement continuous feedback. Students will learn how to recommend and design system feedback mechanisms, implement a process for routing system feedback to development teams, and optimize feedback mechanisms.

Level: Intermediate

Audience

Students in this course are interested in DevOps continuous feedback or in passing the Microsoft Azure DevOps Solutions certification exam.

Prerequisites

- Students should have fundamental knowledge about Azure, version control, Agile software development, and core software development principles. It would be helpful to have experience in an organization that delivers software.
- It is recommended that you have experience working in an IDE, as well as some knowledge of the Azure portal. However, students who may not have a technical background in these technologies, but who are curious about DevOps practices as a culture shift, should be able to follow the procedural and expository explanations of continuous integration regardless.

Expected learning objectives

- Design practices to measure end-user satisfaction
- Design processes to capture and analyze user feedback from external sources
- Design routing for client application crash report data
- Recommend monitoring tools and technologies
- Recommend system and feature usage tracking tools
- Configure crash report integration for client applications
- Develop monitoring and status dashboards
- Implement routing for client application crash report data
- Implement tools to track system usage, feature usage, and flow
- Integrate and configure ticketing systems with development team's work management system
- Analyze alerts to establish a baseline
- Analyze telemetry to establish a baseline
- Perform live site reviews and capture feedback for system outages
- Perform ongoing tuning to reduce meaningless or non-actionable alerts

Syllabus

This course includes content that will help you prepare for the Microsoft Azure DevOps Solution certification exam. Other content is included to ensure you have a complete picture of DevOps. The course content includes a mix of videos, graphics, reference links, module review questions, and hands-on labs.

Module 1 – Recommend and Design System Feedback Mechanisms

- Lesson: The Inner Loop
- Lab: Integration between Azure DevOps and Teams
- Lesson: Continuous Experimentation mindset
- Lab: Feature Flags
- Lesson: Design practices to measure end-user satisfaction
- Lesson: Design processes to capture and analyze user feedback
- Lesson: Design process to automate application analytics

Module 2 – Implement Process for Routing System Feedback to Developmen Teams

- Lesson: Implement tools to track system usage, feature usage, and flow
- Lesson: Implement routing for mobile application crash report data
- Lesson: Develop monitoring and status dashboards
- Lesson: Integrate and configure ticketing systems

Module 3 – Implement and Manage Build Infrastructure

- Lesson: Site Reliability Engineering
- Lesson: Analyze telemetry to establish a baseline
- Lesson: Perform ongoing tuning to reduce meaningless or non-actionable alerts
- Lesson: Analyze alerts to establish a baseline
- Lesson: Blameless PostMortems and a Just Culture

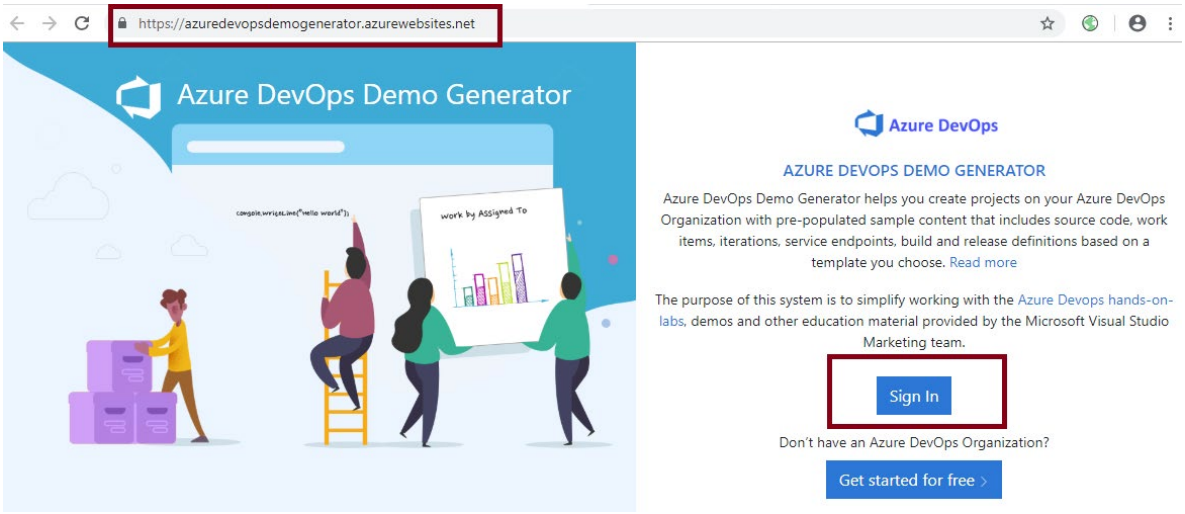
✓ This course uses the **Microsoft DevOps Lab Environment**[2] to provide a hands-on learning environment.

Lab Environment Setup

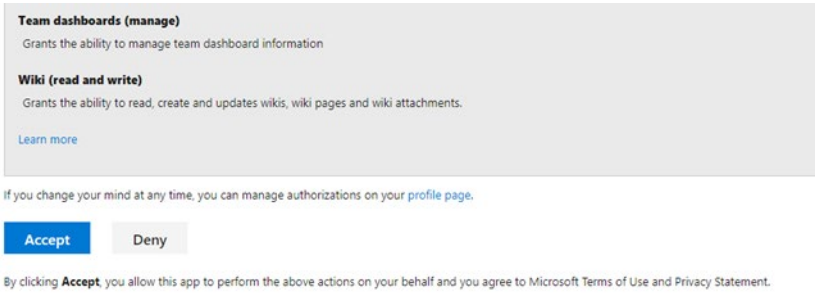
We highly recommend that you complete the assigned hands-on lab work.To do the hands-on labs, you will need to complete the following steps.

1. Sign up for a free **Azure DevOps account**[3]. Use the Sign up for a free account button to create your account. If you already have an account proceed to the next step.
2. Sign up for free **Azure Account**[4]. If you already have an account proceed to the next step.
3. To make it easier to get set up for testing Azure DevOps, a **Azure DevOps Generator Demo**[5] program has been created. Click the **Sign In** button and sign in with your Azure DevOps account.

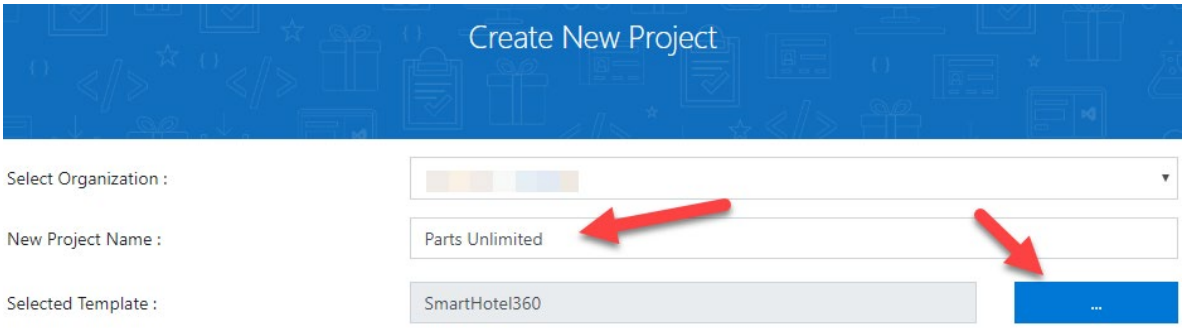
2 https://azuredevopslabs.com/
3 https://www.azuredevopslabs.com/
4 https://azure.microsoft.com/en-us/free/
5 https://azuredevopsdemogenerator.azurewebsites.net/



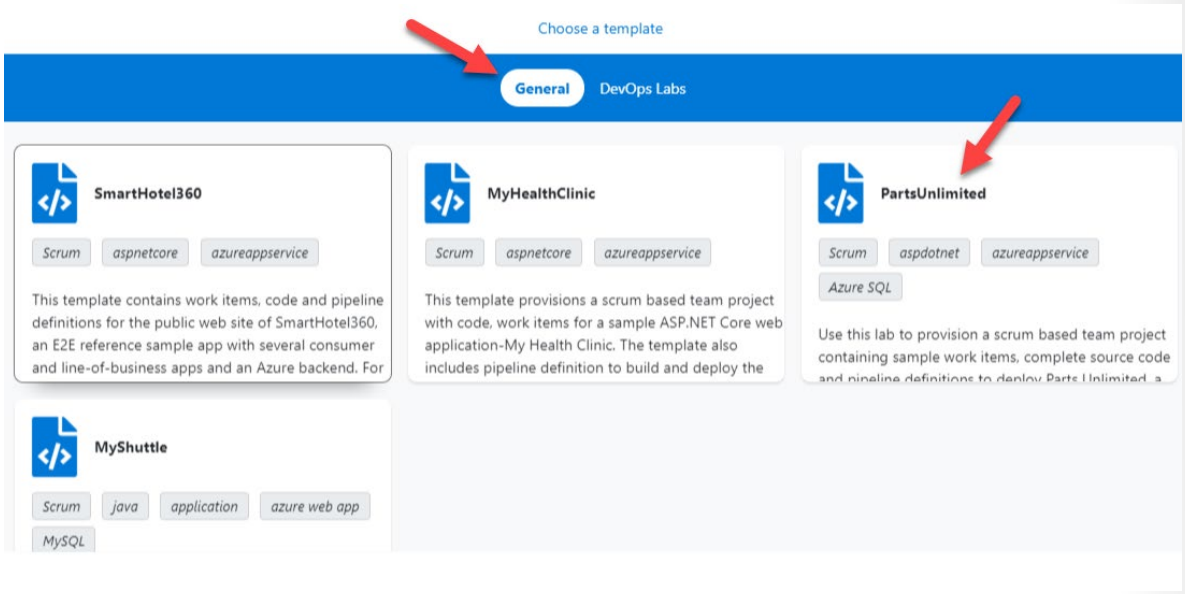
1. You will then be asked to confirm that the generator site can have permission to create objects in your Azure DevOps account.



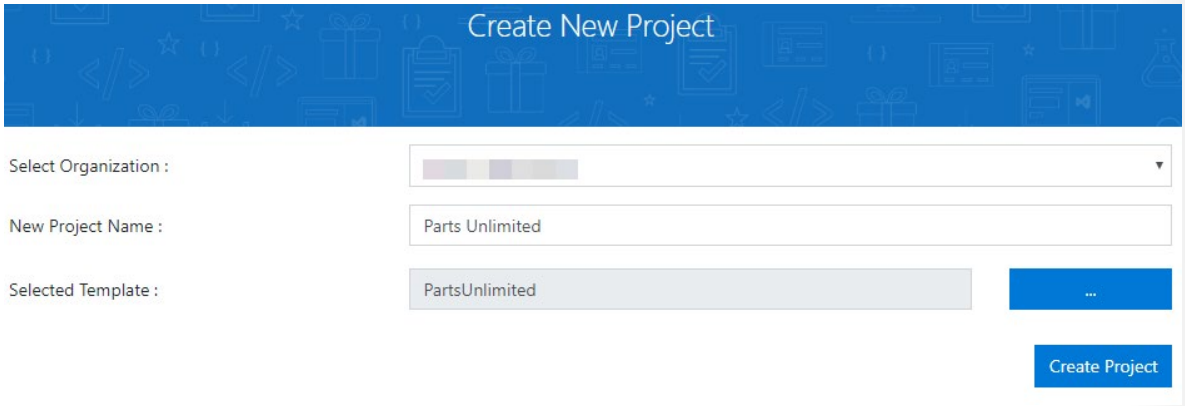
1. If you agree, click the **Accept** button and you should be greeted by the Create New Project screen:



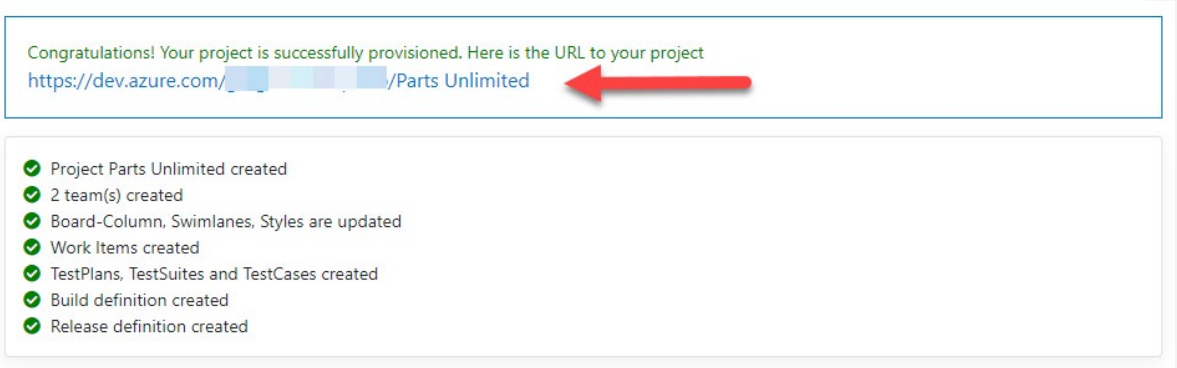
1. Select the appropriate organization (if you have more than one) and enter **Parts Unlimited** as the **New Project Name**, then click the ellipsis to view the available templates. These will change over time but you should see a screen similar to the following:



1. From the **General** tab, choose **PartsUnlimited**, then click **Select Template**.



1. Now that the Create New Project screen is completed, click **Create Project** to begin the project creation phase.



1. When the project is successfully completed, click the link provided to go to your team project within Azure DevOps.

✓ Note that because Azure DevOps was previously called VSTS (Visual Studio Team Services), some of the existing hands-on labs might refer to VSTS rather than Azure DevOps.

Module 1 Recommend and Design System Feedback Mechanisms

The inner loop

Introduction

The ultimate goal of DevOps is to increase the speed and quality of software services, something every IT organization needs. But how can the parties involved in DevOps accelerate delivery if they don't understand what they deliver or know the people to whom they deliver? Despite great advancements in delivery, quality service still begins with understanding the needs of users.

Unsatisfied customers are probably costing you a lot of money. The first step to overcoming this is to admit that you have room for improvement. The second step is to measure customer satisfaction to find out where you currently stand.

Engaging customers throughout your product lifecycle is a primary Agile principle. Empower each team to interact directly with customers on the feature sets they own.

- Continuous feedback: Build in customer feedback loops. These can take many forms:
    - Customer voice: Make it easy for customers to give feedback, add ideas, and vote on next generation features.
    - Product feedback: In-product feedback buttons are another way to solicit feedback about the product experience or specific features.
    - Customer demos: Regularly scheduled demos that solicit feedback from your customers can help shape next generation products and keep you on track to build applications your customers want to consume.
- Early adopter programs: Such programs should be developed with the idea that all teams may want to participate as some point. Early adopters gain access to early versions of working software which they then can provide feedback. Oftentimes, these programs work by turning select feature flags on for an early adopter list.

- Data-driven decisions: Find ways to instrument your product to obtain useful data and that can test various hypotheses. Help drive to an experiment-friendly culture that celebrates learning.

Lab Integration between Azure DevOps and Teams

Microsoft Teams is your chat-centered workspace in Office 365. Software development teams get instant access to everything they need in a dedicated hub for teamwork, that brings your teams, conversations, content and tools from across Office 365 and Azure DevOps together into one place to help improve the inner feedback loop.

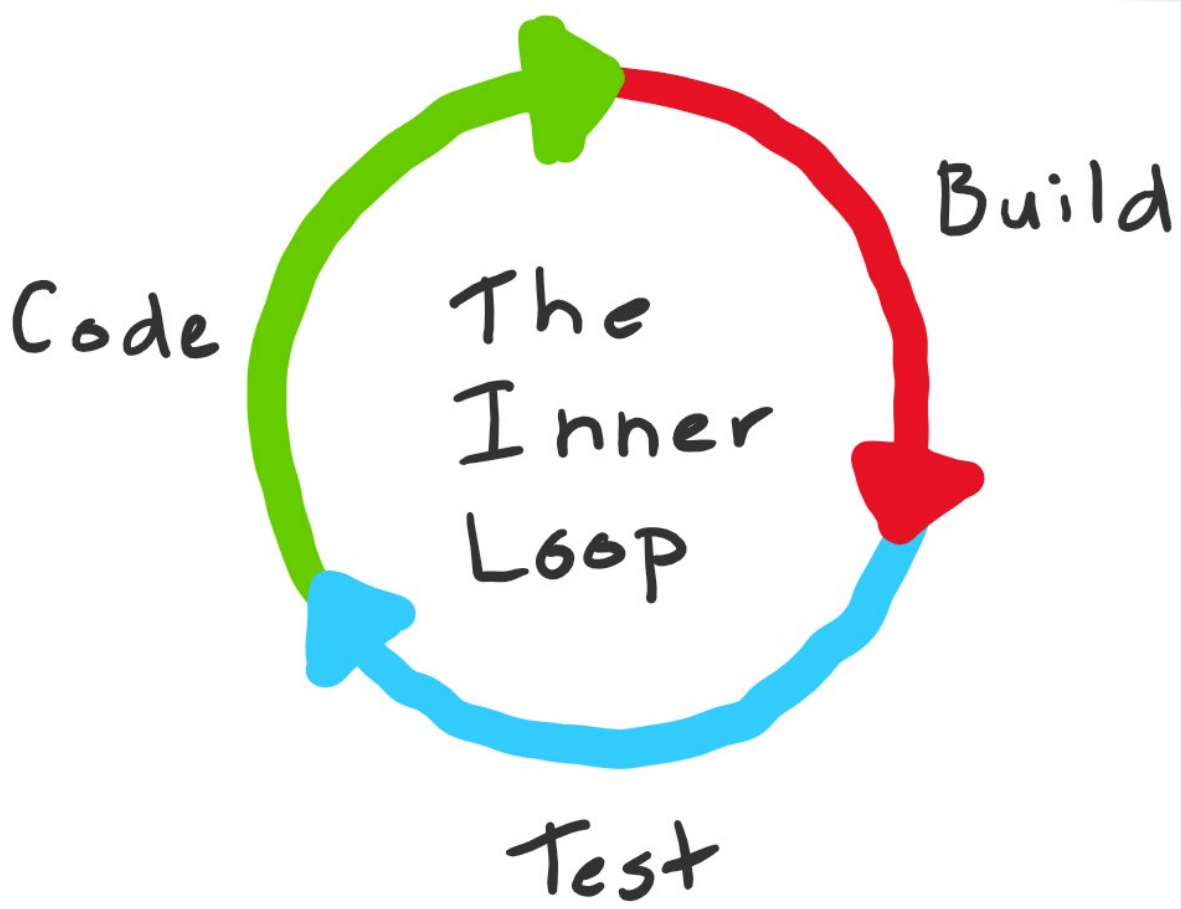
https://azuredevopslabs.com/labs/vsts/teams

The inner loop

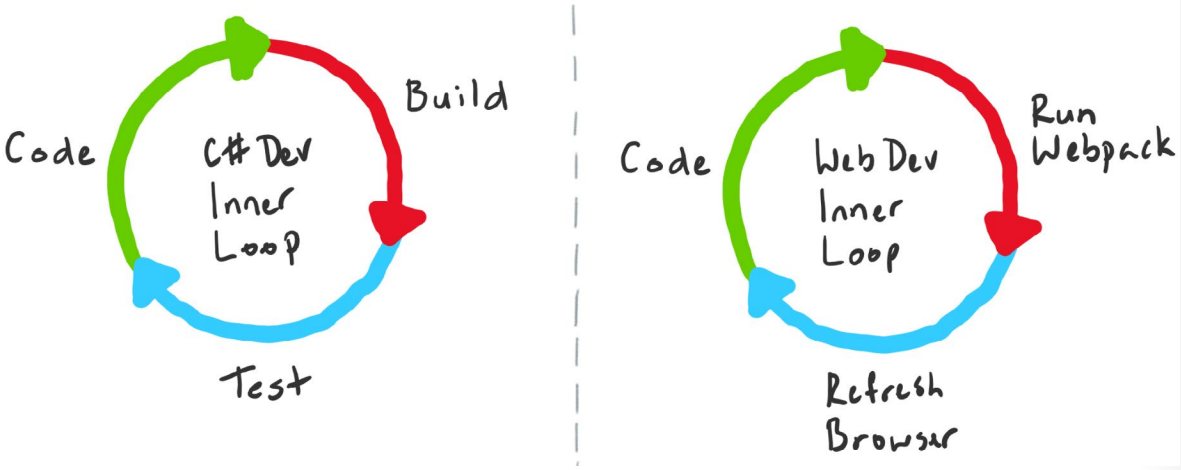
It's not clear who coined the term “inner loop” in the context of software engineering, but within Microsoft at least the term seems to have stuck. Many of the internal teams that I work with see it as something that they want to keep as short as possible - but what is the inner loop?

Definitions

The easiest way to define the inner loop is the iterative process that a developer performs when they write, build, and debug code. There are other things that a developer does, but this is the tight set of steps that are performed over and over before they share their work with their team or the rest of the world.



Exactly what goes into an individual developer's inner loop will depend a great deal on the technologies that they are working with, the tools being used and of course their own preferences. If I was working on a library my inner loop would include coding, build, test execution & debugging with regular commits to my local Git repository. On the other hand if I was doing some web front-end work I would probably be optimized around hacking on HTML & JavaScript, bundling and refreshing the browser (followed by regular commits).



In reality most codebases are comprised of multiple moving parts and so the definition of a developer's inner loop on any single codebase might alternate depending on what is being worked on.

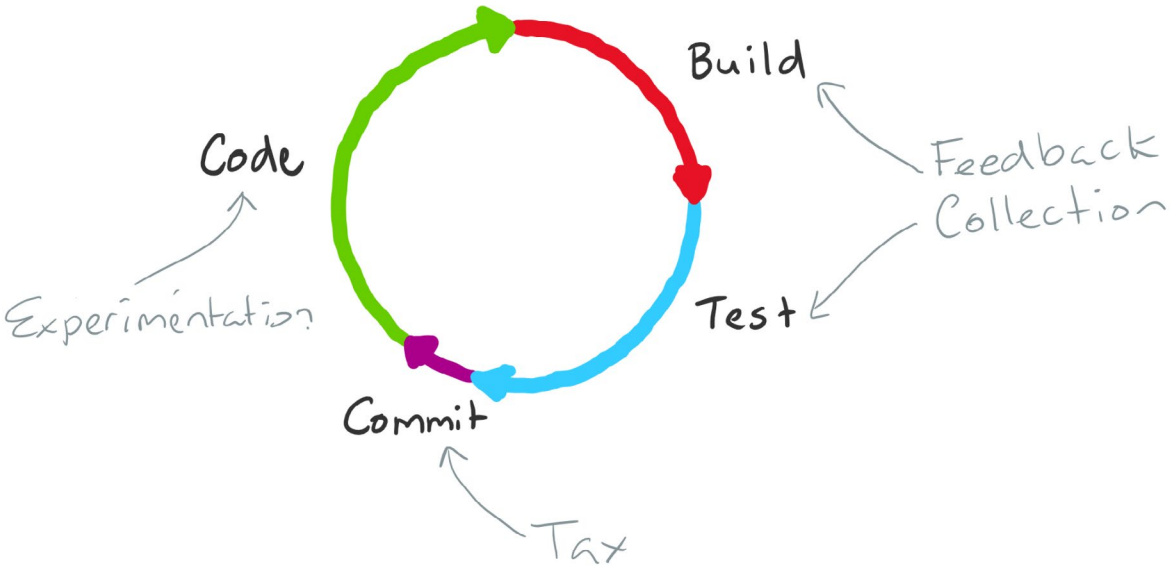
Understanding the Loop

The steps within the inner loop can be grouped into three broad buckets of activity - experimentation, feedback collection and tax. If we flick back to my library development scenario I mentioned earlier, there are four steps I mentioned and how I would bucket them.

- Coding (Experimentation)
- Building (Feedback Collection)
- Testing / Debugging (Feedback Collection)
- Committing (Tax)

Of all the steps in the inner loop, coding is the only one that adds customer value. Building and testing code are important, but ultimately we use them to give the developer feedback about the what they have written to see if it delivers sufficient value (does the code even compile, and does it satisfy the requirements that we've agreed etc).

Putting committing code in the tax bucket is perhaps a bit harsh, but the purpose of the bucket is to call out those activities that neither add value, or provide feedback. Tax is necessary work, if its unnecessary work then it is waste and should be eliminated.



Loop Optimization

Having categorized the steps within the loop it is now possible to make some general statements:

- You want to execute the loop as fast as possible and for the total loop execution time to be proportional to the changes being made.
- You want to minimize the time feedback collection takes, but maximize the quality of the feedback that you get.
- You want to minimize the tax you pay by eliminating it where it isn't necessary on any particular run through the loop (can you defer some operations until you commit for example).
- As new code and more complexity is added to any codebase the amount of outward pressure to increase the size of the inner loop also increases (more code means more tests which in turn means more execution time and a slow execution of the inner loop).

If you have ever worked on a large monolithic codebase it is possible to get into a situation where even small changes require a disproportionate amount of time to execute the feedback collection steps of the inner loop. This is a problem and you should fix it.

There are a number of things that a team can do to optimize the inner loop for larger codebases:

1. Only build and test what was changed.
2. Cache intermediate build results to speed up full builds.
3. Break up the code-base into small units and share binaries.

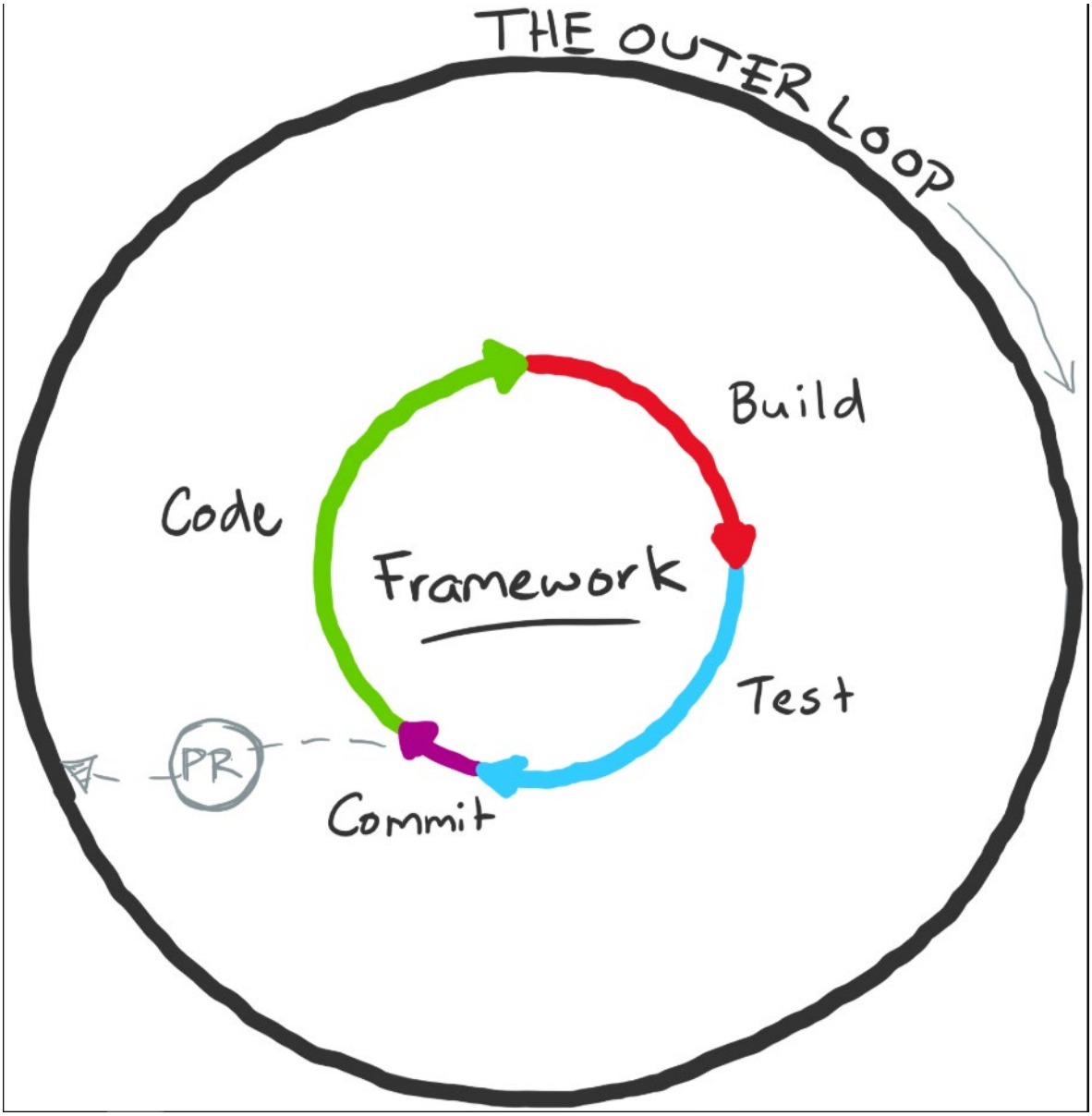
How you tackle each one of those is probably a blog post in their own right. At Microsoft, for some of our truly massive monolithic codebases we are investing quite heavily in #1 and #2 - but #3 requires a special mention because it can be a double edged sword and if done incorrectly and can have the opposite of the desired impact.

Tangled Loops

To understand the problem we need to look beyond the inner loop. Let's say that our monolithic code-base has an application specific framework which does a lot of heavy lifting. It would be tempting to extract that framework into a set of packages.

To do this you would pull that code into a separate repository (optional, but this is generally the way it is done), then setup a separate CI/CD pipeline that builds and publishes the package. This separate build and release pipeline would also be fronted by a separate pull-request process to allow for changes to be inspected before the code is published.

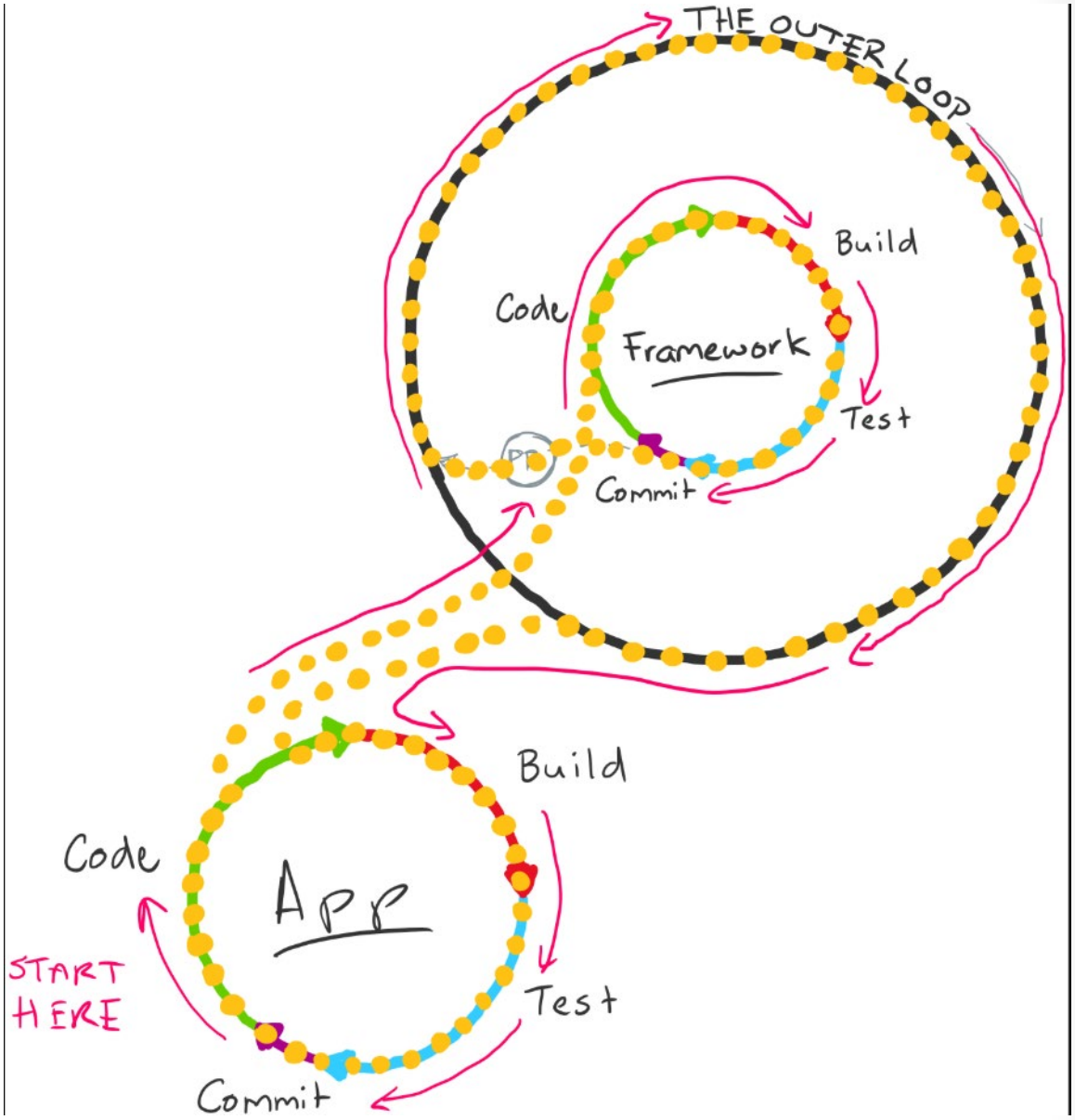
When someone needs to change this framework code they clone down the repository, make their changes (a separate inner loop) and submit a PR which is the transition of the workflow from the inner loop to the outer loop. The framework package would then be available to be pulled into dependent applications (in this case the monolith).



Initially things might work out well, however at some point in the future it is likely that you'll want to develop a new feature in the application that requires extensive new capabilities to be added to the framework. This is where teams that have broken their codebases up in sub-optimal ways will start to feel pain.

If you are having to co-evolve code in two separate repositories where a binary/library dependency is present then you are going to experience some friction. In loop terms - the inner loop of the original codebase now (temporarily at least) includes the outer loop of the framework code that was previously broken out.

Outer loops include a lot of tax such as code reviews, scanning passes, binary signing, release pipelines and approvals. You don't want to pay that every time you've added a method to an class in the framework and now want to use it in your application.



What generally ends up happening next is a series of local hacks by the developer to try and stitch the inner loops together so that they can move forward efficiently - but it gets pretty messy quick and you have to pay that outer loop tax at some point.

This isn't to say that breaking code up into separate packages is an inherently bad thing - it can work brilliantly, you just need to make those incisions carefully.

Closing Thoughts

There is no silver bullet solution that will ensure that your inner loop doesn't start slowing down, but it is important to understand when it starts happening, what the cause is and work to address it.

Decisions such as how you build, test and debug, to the actual architecture itself will all impact how productive developers are. Improving one aspect will often cause issues in another.

Continuous Experimentation mindset

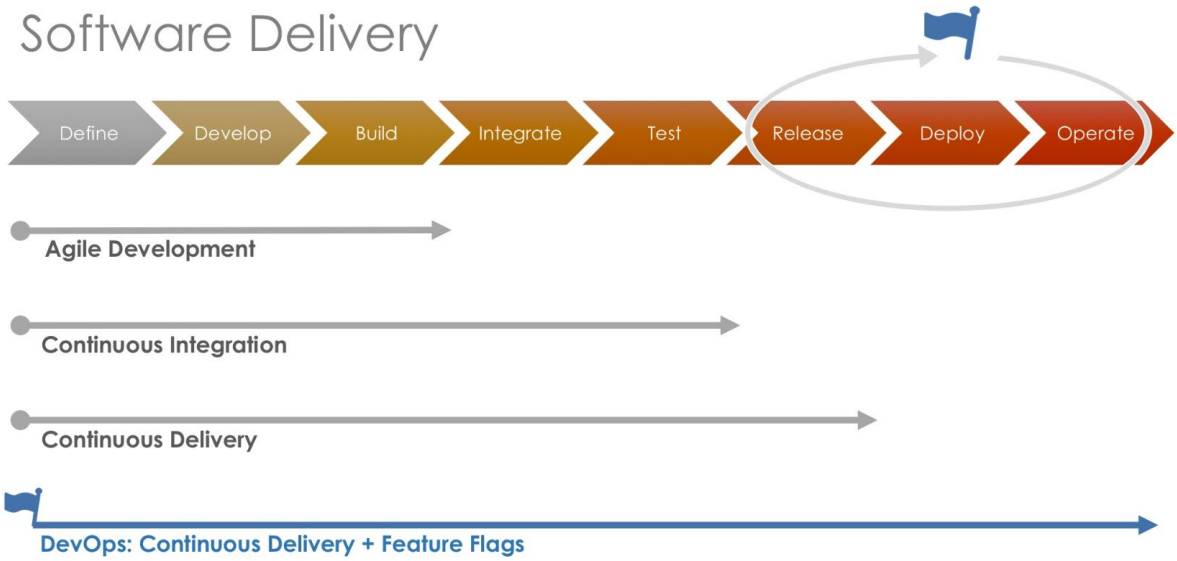
Continuous Experimentation mindset

We are in the era of continuous delivery, where we are expected to quickly deliver software that is stable and performant. We see development teams embracing a suite of continuous integration/delivery tools to automate their testing and QA, all while deploying at an accelerated cadence.

However, no matter how hard we try to mitigate the risk of software delivery, almost all end-user software releases are strictly coupled with some form of code deployment. This means that companies must rely on testing and QA to identify all issues before a release hits production. It also means that companies primarily rely on version control systems or scraped together config files to control feature releases and mitigate risk. Once a release is in production, it is basically out in the wild. Without proper controls, rolling back to previous versions becomes a code deployment exercise, requiring engineering expertise and increasing the potential for downtime.

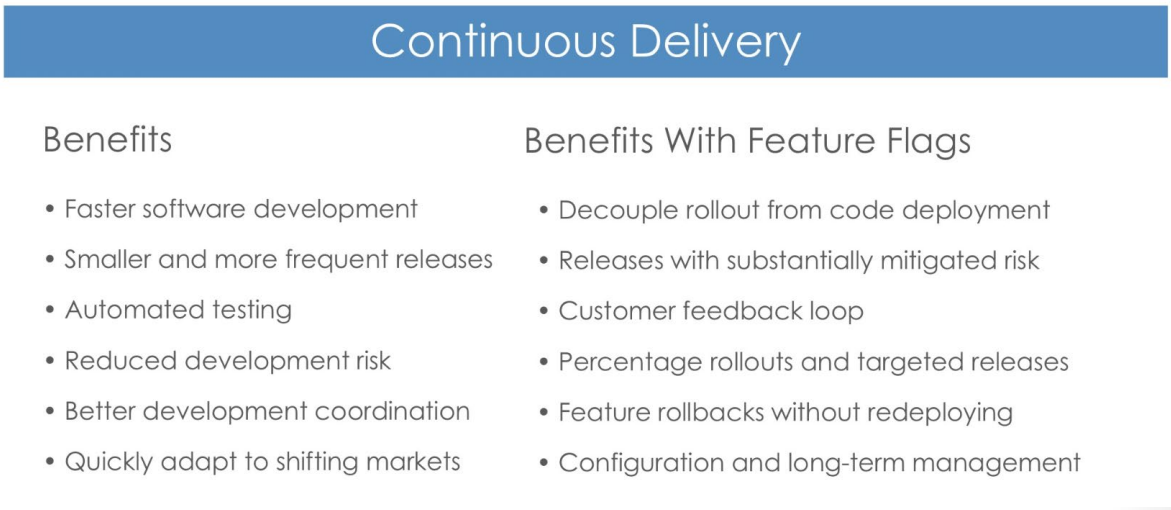
One way to mitigate risk in feature releases is to introduce **feature flags**[1](feature toggles) into the continuous delivery process. These flags allow features (or any code segment) to be turned on or off for particular users. Feature Flags are a powerful technique, allowing teams to modify system behavior without changing code.

In this new reality, software development and operations capabilities set the boundaries for success. The first step to creating a company-wide culture of experimentation is for executives, the business, developers, and ops–and other key stakeholders–to understand how changes in engineering productivity align with business outcomes. Innovation is the key to success, and success depends on hypothesis-testing through experimentation. By adopting a culture of continuous experimentation, features can be tested by creating an instrumented minimal viable product rapidly and release to a subset of customers in production for testing, this enables the team to make fact based decisions and quickly evolve towards an optimal solution.



By using feature flags in the continuous delivery process, teams are able to efficiently integrate release, deployment, and operational management into the software development cycle.

1 http://blog.launchdarkly.com/feature-flag-driven-development/



From their onset, feature flagging platforms have accelerated the progression of DevOps practices. Feature flags have always fit into DevOps practices due to the increased control over the delivery cycle that they offer, but the mitigation of risk and prevention of associated technical debt has brought flagging platforms to the spotlight as quintessentially "DevOps". These changes alongside the newly offered ability to monitor all changes to a flag via an audit log have widened the reach of the methodology. Simultaneously, these systems deeply supplement the methodology as it currently exists and provide additional use cases and benefits that push the limits of DevOps beyond what was possible before.

Using Feature Flags to Safely Get Feedback on Features

We are in an era of continuous delivery, where we are expected to quickly deliver software that is stable and performant. We see software development teams embracing a suite of continuous integration/delivery tools to automate their testing and QA, all while deploying at an accelerated cadence. No matter how hard we try to mitigate the risk of software delivery, almost all end-user software releases are strictly coupled with some form of code deployment. This means that companies must rely on testing and QA to identify all issues before a release hits production. There are two key challenges when testing features in test environments:

- Testing in test environments can be challenging if your test scenarios depend on production quality data. It can take a lot of effort to create this kind of data in test environments and it's likely you'll still miss out on key test scenarios, since in some cases the effort involved in creating this data out weights the benefits.
- The other most common scenario is doing user testing, inspecting, and adapting the functionality of your product based on usage data. End users may be invested in the success of your product, but it can get increasingly difficult to get constant feedback on every functionality in a test environment.

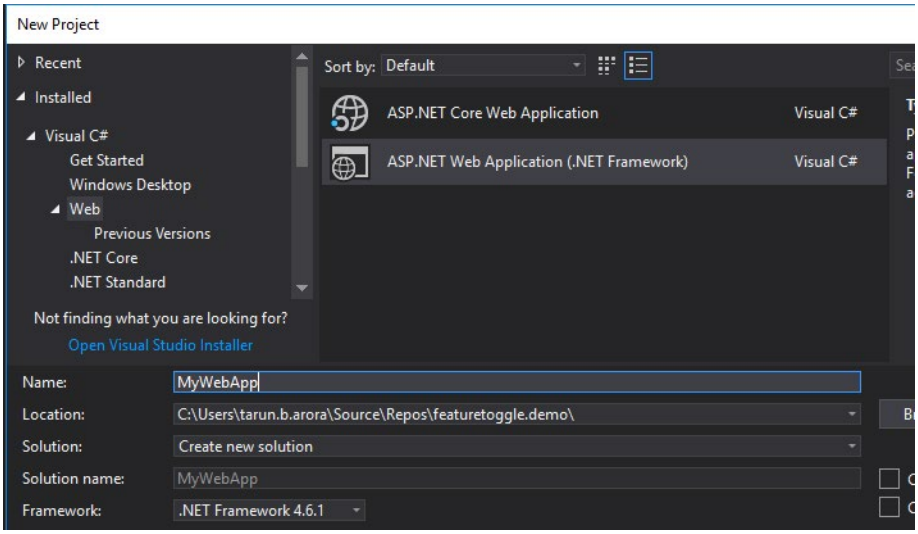
Once a release is in production, it is basically out in the wild. Without proper controls, rolling back to previous versions becomes a code deployment exercise, requiring engineering expertise and increasing the potential for downtime. One way to mitigate risk in feature releases is to introduce feature flags (feature toggles) into the continuous delivery process. These flags allow features (or any code segment) to be turned on or off for particular users. Feature flags are a powerful technique, allowing teams to modify system behaviour without changing code.

Innovation is the key to success, and success depends on hypothesis-testing through experimentation. By adopting a culture of continuous experimentation, features can be tested by creating an instrumented minimal viable product rapidly and released to a subset of customers in production for testing; this enables the team to make fact-based decisions and quickly evolve towards an optimal solution.

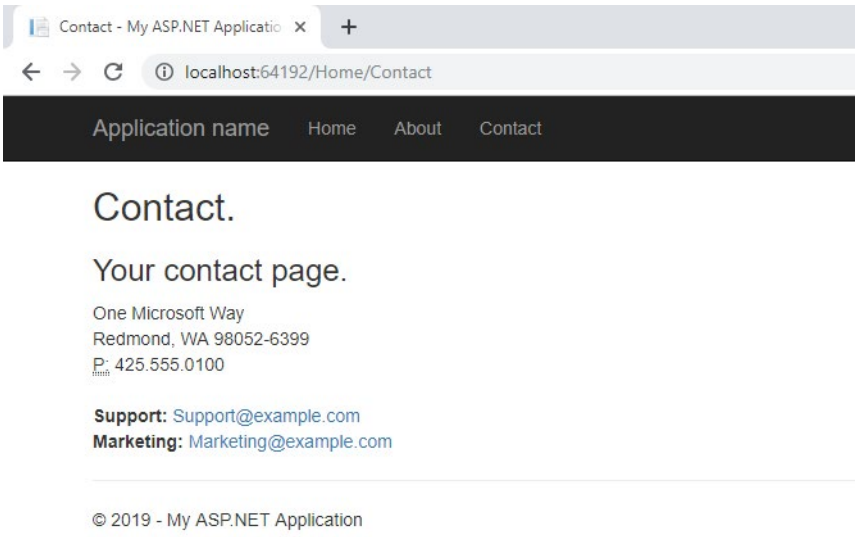
In this tutorial we'll learn how to get into a true continuous testing culture by leveraging feature flags.

Getting ready

1. Create a new web application using the ASP.NET Web Application template in Visual Studio, name it MyWebApp, and save it in a new folder called featuretoggle.demo:



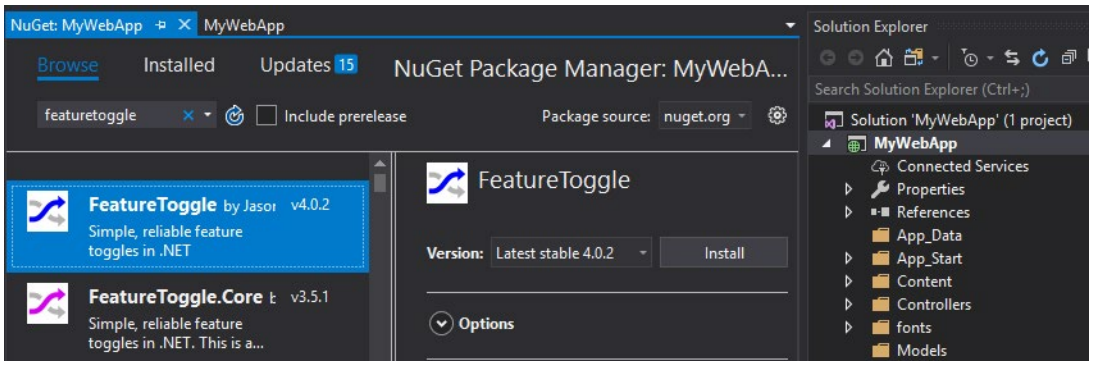
1. Simply build and run the website, then navigate to the Contact form:



In the next section, we'll see how to use feature flags to deploy changes to the Contact form without releasing the changes to everyone.

How to do it

1. In the MyWebApp project, add a reference to the FeatureToggle package



1. Next, create a folder called Toggle and add a class called `NewContactForm.cs`. Copy the following code into this class file

```
using FeatureToggle;
namespace MyWebApp.Toggle
{    public class NewContactForm : SimpleFeatureToggle
     {
     }
}
```

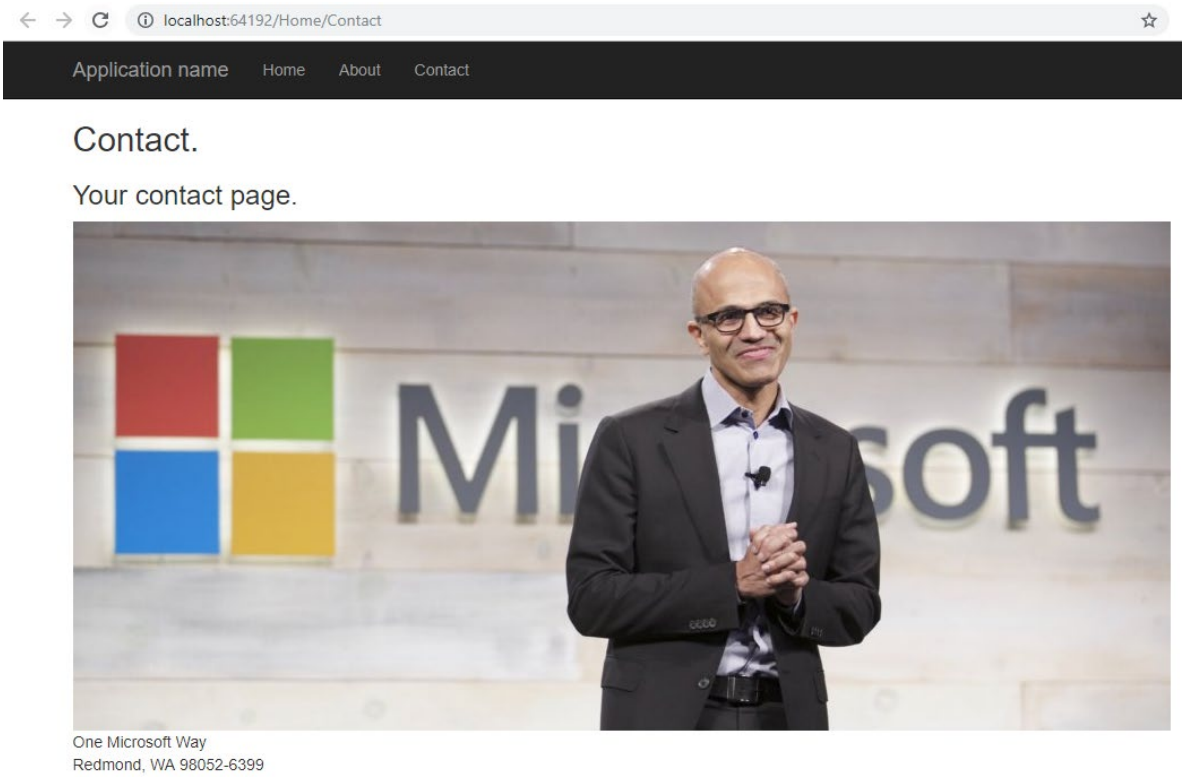
1. Add a new app key in the web.config file, set the key name to. FeatureToggle.NewContactForm and the value to false. This key will be used to control the feature flag:

```
<appSettings>
  <add key="webpages:Version" value="3.0.0.0" />
  <add key="webpages:Enabled" value="false" />
  <add key="ClientValidationEnabled" value="true" />
  <add key="UnobtrusiveJavaScriptEnabled" value="true" />
  <add key="FeatureToggle.NewContactForm" value="false"/>
</appSettings>
```

1. Next, modify the `Contact.cshtml` page under Views\Home to include the following code:

```
@{ var toggle = new MyWebApp.Toggle.NewContactForm();
if (toggle.FeatureEnabled)
{
      <img src="https://www.incimages.com/uploaded_files/image/970x450/
getty_459885938 _144096.jpg" />
}
}
```

1. Build and run the project. Navigate to the Contact form page and you'll notice it's unchanged. Update the value of the FeatureToggle.NewContactForm key in the web.config file from false to true.
2. Now refresh the Contact form. You'll see the updated page with the image:



How it works

The feature toggle package includes a series of providers that can be used to control the value of an object that can, in turn, be used to decide whether the feature is accessible. You may ask why we use feature toggle. Well, it is easy to construct a simple if....else condition using a config key to control when the page gets shown. While magic strings can be used, toggles should be real things (objects), not just a loosely typed string. This helps effectively manage the feature flags overtime. When using real toggles you can do the following:

- Find uses of the Toggle class to see where it's used
- Delete the Toggle class and see where a build fails

Feature flags allow you to decouple code deployments from feature releases. This simplifies testing code changes in production without impacting end users. By using feature flags it's possible to control who can see a feature; it's also possible to phase in traffic to a new feature rather than opening up all users at once. You can read more about feature flags and their benefits **here**[2]

There's more

The feature toggle package also provides the following feature toggle types:

- AlwaysOffFeatureToggle
- AlwaysOnFeatureToggle
- EnabledOnOrAfterDateFeatureToggle

2 https://martinfowler.com/articles/feature-toggles.html

- EnabledOnOrBeforeDateFeatureToggle
- EnabledBetweenDatesFeatureToggle
- SimpleFeatureToggle
- RandomFeatureToggle
- EnabledOnDaysOfWeekFeatureToggle
- SqlFeatureToggle
- EnabledOnOrAfterAssemblyVersionWhereToggleIsDefinedToggle

More details on these feature toggle types and their usage can be found **here**[3]

Lab Feature Flags

In this lab, you will learn:

- How to implement a very simple feature flag for an ASP.NET MVC application.
- How to integrate LaunchDarkly with Azure DevOps
- How to roll-out LaunchDarkly feature flags in Azure DevOps release pipelines

https://www.azuredevopslabs.com/labs/vstsextend/launchdarkly

3 http://jason-roberts.github.io/FeatureToggle.Docs/pages/usage.html

Design practices to measure end-user satisfaction

Design practices to measure end user satisfaction

"Measurement is the first step that leads to control and eventually to improvement. If you can't measure something, you can't understand it. If you can't understand it, you can't control it. If you can't control it, you can't improve it." — H. James Harrington

'User experience' encompasses all aspects of the end-user's interaction with the company, its services, and its products. UI (user interface) is not UX (user experience). A car with all its looks, dashboard and steering wheel is the UI. Driving it is the UX. So the interface directly contributes to the experience (beautiful car interior makes a better experience sitting in one), but is not the experience itself.

The only real measure of software quality that's worth its salt is end-user satisfaction. Forget about what applications have been designed to do; forget about developer and service provider promises that don't include an end-user analytical strategy. Cut to the chase and look at what's actually happening; Measuring customer satisfaction doesn't have to be complicated or expensive. In fact, it's fairly simple to incorporate customer satisfaction measurement into your current customer success strategy. No matter how you cut it, measuring customer satisfaction comes down to gathering customer feedback via surveys. To accurately gauge customer sentiment, we'll simply need to ask them how their experience was.

Of course, there are multiple ways you can execute said survey, from the survey design to timing, sample, and even how you analyze the data. Let's briefly cover the five steps to easily measuring customer satisfaction.

1. Outline your goals, and make a plan
When embarking on any sort of campaign, it's helpful to take a step back and ask, "Why are we doing this?" In business, one must weigh the value of additional information (i.e. the customer satisfaction data) in relation to the cost of collecting it (i.e. the survey process). To be honest, if you won't change anything after collecting your customer satisfaction data, you're better off not collecting it. It's going to take time and effort, so you need to put it to use.

Depending on your business or organizational capabilities, there is a lot you can do with the information. One use is simply to wake you up to the reality of any business: A portion of your customers is going to have an unsatisfactory experience. Every business faces this problem.

When you wake up to that fact, you can choose from many routes to correction. You can:

- Improve key UX bottlenecks that contribute to poor customer experience.
- Expedite customer support interactions with the most frustrated customers.
- Operationalize proactive support like a knowledge base and customer education.
- Test different live chat scripts and support strategies.

The specific solution isn't necessarily the important part here. The important part is stepping back and saying, "If we see that a segment of our customers is unsatisfied, what will we do about it?"

2. Create a customer satisfaction survey.
What types of metrics measure customer satisfaction? You can choose among a few different options for

customer satisfaction surveys. There's no unanimous agreement on which one is best. A few popular methods are:

- Customer Satisfaction Score (CSAT)
- Customer Effort Score (CES)
- Net Promoter Score® (NPS)

These are all "one-question" methods that vastly simplify the process of collecting customer insights.

While you may not think the survey methodology matters much, how you ask the question does seem to measure slightly different things.

For instance, a 2010 study found twenty percent of "satisfied" customers said they intended to leave the company in question; 28% of the "dissatisfied" customers intended to stay. So "satisfied" doesn't necessarily equate to "loyal."

- Customer Satisfaction Score (CSAT) is the most commonly used satisfaction method. You just ask your customers to rate their satisfaction on a linear scale. Your survey scale can be 1 – 3, 1 – 5, or 1 – 10, and there is no universal agreement on which scale is best to use.
  CSAT is a metric used to immediately evaluate a customer's specific experience. "CSAT is a transactional metric that is based on what's happening now to a user's satisfaction with a product or service."
- Customer Effort Score (CES) is very similar, but instead of asking how satisfied the customer was, you ask them to gauge the ease of their experience. You're still measuring satisfaction, but in this way, you're gauging effort (the assumption being that the easier it is to complete a task, the better the experience). As it turns out, making an experience a low-effort one is one of the greatest ways to reduce frustration and disloyalty.
- Net Promoter Score (NPS) asks the question, "How likely is it that you would recommend this company to a friend or colleague?" This attempts to measure customer satisfaction but also customer loyalty. In doing so, you can come up with an aggregate score, but you can also easily segment your responses into three categories: detractors, passives, and promoters. You calculate your Net Promoter Score by subtracting the percentage of Detractors from the percentage of Promoters.

You can, of course, use more than one methodology, as well (since they all measure something very slightly different). You can optionally combine multiple scores for a greater picture:

For example, a customer that has had 3 continuous negative CSAT scores over a period and is also a detractor on NPS would be an immediate at-risk customer, while a customer with positive CSAT and a promoter on NPS are potentially the best source of advocates & candidates to cross-sell/upsell since they already have seen the value in their interactions with the process & product.

In addition, I recommend always appending a qualitative open-ended question, regardless of the customer satisfaction survey you use. Without an open-ended question, you risk limiting your insight into "why" the dissatisfaction may be occurring. Qualitative user feedback can give you tons of ideas when it comes to implementing solutions.

3. Choose your survey's trigger and timing.
This step is all about to whom you send the survey and when you send it. If you go back to your goals outline, this shouldn't be too hard to determine, at least strategically. People tend to forget this step, though, but it's of crucial importance and affects the quality and utility of your data. Tactically, you can trigger a survey pretty much anywhere at any time and to anyone nowadays, but strategically, it matters specifically when and where.

"Although there is no "one size fits all" approach to customer satisfaction surveys, there are 3 factors that every company should consider before surveying: What event or action took place before you asked for

feedback (these can be time or action based events like completing your onboarding campaign), the time since your last survey to the customer, and is your team's ability to reply to feedback in a timely manner.

Good examples of event data that can be used to fire a survey are:

- Time since signup
- Key actions taken in your app
- Complete user on-boarding
- Surveying too often will result in low response rates, we recommend a customer satisfaction (NPS) survey seven days after signup, 30 days after the first survey and every 90 days during the customer lifecycle."

With all the options for triggering, though, let's start with some best practices:

- The closer the survey is to the experience, the better.
- People forget how they felt the longer you wait.
- Who you survey changes what insights you get. If you survey website visitors about their satisfaction, the respondents are anonymous and may be a customer – or may not be. This will bring you different data than sending an email to recent customers will. Keep that in mind.
- You should survey your customers more than once to see how things change longitudinally. Especially if you operate a SaaS company or a subscription service, regular NPS surveys can help you analyze trends both at the aggregate and individual level.
- Survey people after a key moment of their customer journey.
- If a respondent gives you a high score, think about adding a follow-up ask. For instance, Tinder asks you to rate their app in the app store if you give them a high score.

4. Analyze the survey data.
Once you've collected your data, make sure it doesn't just sit there dormant and unused. You've got all this customer insight, and it's just waiting to be uncovered!

5. Make adjustments and repeat.
Back to my first point: Now that you have these insights, what are you going to do about it? Ultimately, this is a personal decision that will reflect your own findings and capabilities. You may find that a whole segment is dissatisfied because of a particular experience. In that case, you may need to further investigate why that experience (or product) is causing dissatisfaction and run experiments to try to improve upon it. You may find that you have a small percentage of super fans.

Now that you can identify these people, perhaps you can work with your customer marketing and customer success teams to plan advocacy programs.

The possibilities are endless, but it all starts with accurately measuring customer satisfaction. But asking for scores is only a part of it – make sure you're creating conditions for customers to leave you high scores, too.

In product feedback

It’s very likely that if you visit a web property for a little while, you’ll eventually be prompted to provide some feedback or sign up for something. Rate the app. Complete a one-question survey… Send a smile/frown, shake the device to provide feedback and various other interesting ways to engage users for feedback.

It is about meeting your customers where they are, done well, **in-product** feedback is a fantastic way to gather impactful, relevant information from people as they use your product. Done poorly, it's a lethally effective way to drive people away or irritate them so much that they want to exact their revenge.

What kind of feedback can you expect?

Feedback from people who are already using your products, who have chosen to visit your website, who have otherwise opted-in to what you are selling…feedback from these people is worth its weight in pixels. Providing them a way to share feedback with you from within the context of your product makes this type of feedback especially valuable. It may be used to improve the usability of a particular feature or to recruit users who are already performing certain actions for deeper discussions. The main value is the immediacy of the feedback given the user's context within your product. The feedback will tend to be more concrete, and users may be more likely to provide candid, quick responses.

Benefits

- Context-Sensitive Feedback. Users will already be using your product, so they can provide feedback based on their actual usage or needs at the time. Priceless.
- Always on. By implementing feedback mechanisms within the product itself, you've made it very easy for users to provide input, at any point, without sending a formal survey or otherwise cluttering an inbox in hopes of getting a hit.
- High response rates. Since the feedback mechanism is built into your services, users are able to access it if and when they need it. That could mean reporting a problem, bug, enhancement or glitch or complementing the team on their choice of user experience.

Weaknesses

- Too. Much. Feedback. There are a lot of channels which users can tap into to provide feedback. Sometimes, there are just too many to stay on top of them all. After all, it would be a shame to collect feedback from multiple channels and not have means to review it all…
- Not enough detail. If you are posting micro-surveys within your site, the information you get back from respondents may not be sufficiently detailed to allow it to be actionable.
- Always on. By implementing feedback mechanisms within the product itself, you've made it very easy for users to provide input, at any point, without sending a formal survey or otherwise cluttering an inbox in hopes of getting a hit. But, sometimes that feedback may be irrelevant given new decisions.
- Limited future follow-up. Depending on the tools being used, you may not have enough contact information to follow-up directly with the person who submitted the feedback and delve deeper into their responses.

Tapping into In-Product feedback is helpful throughout the Product Development Lifecycle… In spite of it's weaknesses if done right, In-app feedback is an excellent way to validate existing or proposed functionality, and to solicit ideas for improvements to the status quo.
Once the product is in production, use in-app tools to support users, allowing them to report issues, frustrations, improvements, and enhancements.

If you sell a software product, asking for feedback directly inside the app is a fantastic method for collecting product feedback.

It helps you narrow in on specific issues your customers are experiencing. However, it can also feel like paradox of choice since you can ask ANY question. Here are a few example questions that may be helpful to ask:

"What is {insert product feature} helping you accomplish?"
"What issues, if any, are you having with {insert product feature}?"
"What features do you think we're missing today for {insert product feature}?"

There are hundreds of in-app questions you can ask. Here's a preview of the pros and cons for in-app surveys.

Pros	Cons
Lots of flexibility - you can ask whichever question you see fit, whether you're evaluating a new design, gauging how customers feel about a new feature launch, etc.	Difficult to comb through open-ended responses and extract insights.
Gives us access to the customer/user where they are in the app.	Low response rates.
Gives us context on what the user/customer is looking at in the app right before their response.	No ability to throttle NPS based on if a user has recently responded to feedback – need to be able to suppress certain users.
Allows us to respond in-app so I can keep all of my feedback in one place.	

Feedback on product roadmap

The big thing that gets missed from feedback surveys is priority. Wouldn't it be great if there was a way to capture feedback and allow your customers to vote on what is most useful to them?

Effective communication on your product roadmap helps keep your customers engaged.. However, by giving your users the ability to request for features you can make them part of your product development inner loop. This is powerful, but at the same time you run the risk of driving your product into a direction where it becomes a niche solution only for a subset of the envisaged target market. By increasing the visibility on the requested features you can empower your full customer base to vote on features they would like to see most. This helps add a third dimension to product backlog and aids prioritisation.

It's been shown that increased communication with customers promotes more engagement and feedback. By updating supporters of an idea as it moves through different product development stages - from discovery to planning to development and launch - your customers never feel like they're in the dark.

The Azure DevOps team uses public projects to show it's feature plan for the next few months, you can check it out **here**[4].

Pros	Cons
Customers feel that they're an active part of building your product roadmap. It provides a place to make customers feel their voice is heard	Likely biased towards your highest-intent customers. The people who aren't using your product are much more likely to withhold feedback or product suggestions.

4 https://dev.azure.com/mseng/Azure%20DevOps%20Roadmap/_workitems/recentlyupdated

Pros	Cons
Builds a sense of community and heightened loyalty when you can collaborate with the company on ideas.	Low volume unless customers are explicitly prompted to suggest an idea in the board.
Provides a channel through which you can make users feel appreciated for their contributions by letting them know that you're taking action on their suggestions.	

Feature Request Board

A massive part of build a product is identifying new features customers desire. The easiest way to figure it out? Ask them! Creating a "feature request board" is a common tool for gauging product feedback from existing customers. Let's see how the integration between User Voice and Azure DevOps can help you create your own feature board for opening the feedback channels with your customers…

https://feedback.uservoice.com/knowledgebase/articles/363410-vsts-azure-devops-integration

Design processes to capture and analyze user feedback

Introduction

"People are talking about you and your competitors constantly. They could be asking for help, complaining about a bug, or raving about how much they love you. And you want to stay on top of all those conversations. Your customers' opinions matter, not just to PR & marketing, but to every team – from customer support to product development to sales. But sifting through irrelevant posts on multiple channels is overwhelming and can be a huge time drain. Sometimes it's not even possible when they don't tag or link to you. That's where scanning tools comes in. Scanning tools make it easy for you to find people talking about you, but not necessarily to you (when they don't @mention your account) - and reach out or take notes when necessary. There are so many business opportunities to uncover if you know where to look."

Although there isn't a specific survey question you can ask … it's important to get a general pulse on what your customers are saying over time about your company. Here's the pros and cons:

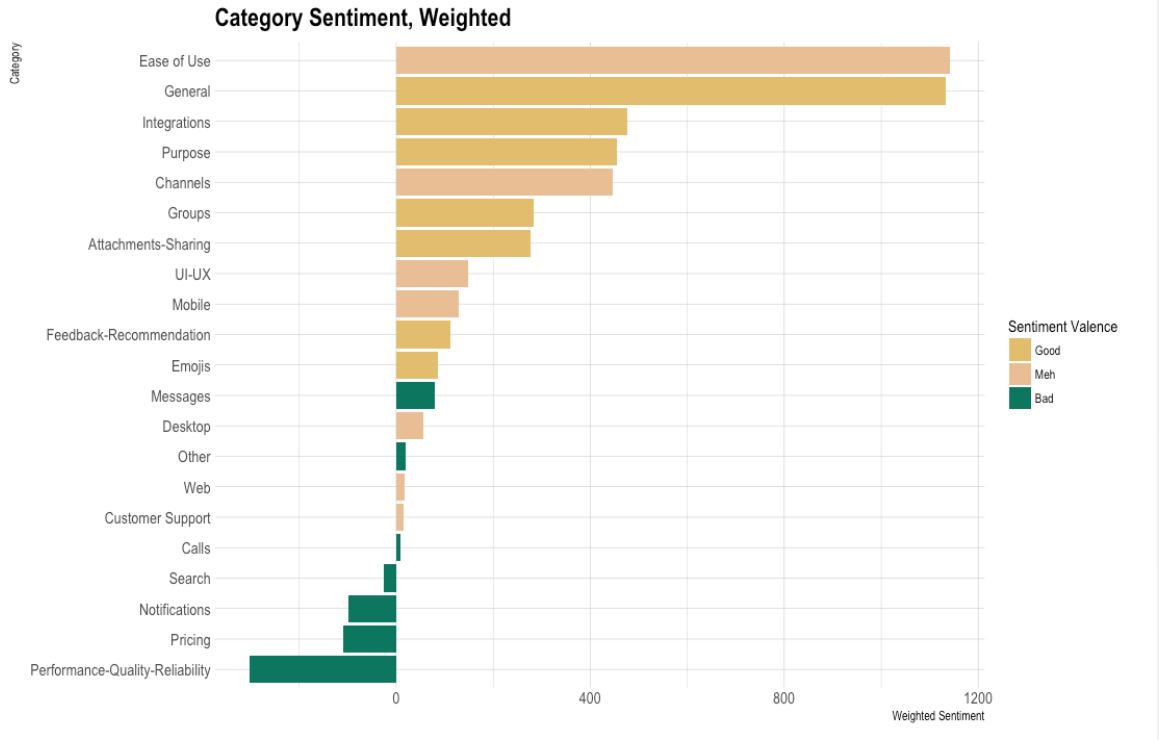
Pros	Cons
No burden on the customer to complete a survey. In their natural environment.	Difficult to measure and quantify which makes it nearly impossible to track performance over time.
Get a true measure of what customers think about you as this method is entirely organic.	Challenging to tie social media comments back to a CRM system at scale.

Customer feedback doesn't just come in through your site's contact form – it's everywhere. You only have to search the Twitter handle of any product with more than a few hundred users to see that customers love to offer their opinion – positive and negative. It's useful to be monitoring this and learning from it, but casually collecting feedback on an ad-hoc basis isn't enough.

Startups thrive on feedback as their 'North star', and are constantly evolving based on what their customers request, break, and complain about. Enterprises also can't overlook the fact that customers are what make any company tick, and must struggle harder than startups to stay relevant and innovate.

So, if you're just collecting feedback 'as and when' it comes in, you're missing out on data that's just as important as page views or engagement. It's like deciding not to bother setting up Google Analytics on your homepage, or not properly configuring your CRM; in the end, you're deciding to not benefit from data that will have a transformative effect on your product strategy. With a dataset of feedback – whether that's from customer reviews, support tickets, or social media – you can dig into the words your customers are using to describe certain parts of your product and get insights into what they like, and what they don't like.

Here's the kind of end result you can get with this



The outcome of AI analysis on Slack reviews. The categories on the left refer to different parts of Slack's product, and the bars represent how positively or negatively customers feel about each.

As the saying goes, "For every customer who bothers to complain, 20 other customers remain silent."

Unless the experience is really bad, customers usually don't bother to share feedback about an experience that didn't meet their expectations. Instead, they decide never to do business with the service provider again. That's a high price to pay for lost feedback.

An excellent source of feedback is on other websites, such as online communities, blogs, local listings, and so on. If your customers are not happy with the resolution to a negative experience, they are likely to vent their ire on these forums. The lost customer is not the only casualty. Studies have shown that each dissatisfied customer typically shares the unsatisfactory experience with 8 to 10 (sometimes even 20) others. With the growing use of social media, it's not uncommon for negative feedback to go viral and hurt the credibility of a brand.

Release Gates

Any responsible DevOps practice uses techniques to limit the damage done by bugs that get deployed into production. One of the common techniques is to break up a production environment into a set of separate instances of an app and then configure deployments to only update one instance at a time, with a waiting period between them. During that waiting period, you watch for any signs (telemetry, customer complaints, etc.) that there is a problem and if so, halt the deployment, fix the issue and then continue the deployment. This way, any bug you deploy only affects a small fraction of your user base. In fact, often, the first product environment in the sequence is often one only available to internal people in your organization so you can validate the changes before they hit "real" customers. None-the-less, sometimes issues make it through.

Release gates automate the waiting period between environments in release pipelines. They enable you to configure conditions that will cause the release wait. Out of the box, a few conditions are supported namely Azure monitoring alerts and Work item queries. Using the first, you can have your release hold if your monitoring alerts are indicating that the environments you've already deployed to are unhealthy. And the second allows you to automatically pause releases if anyone files a "blocking bug" against the release.

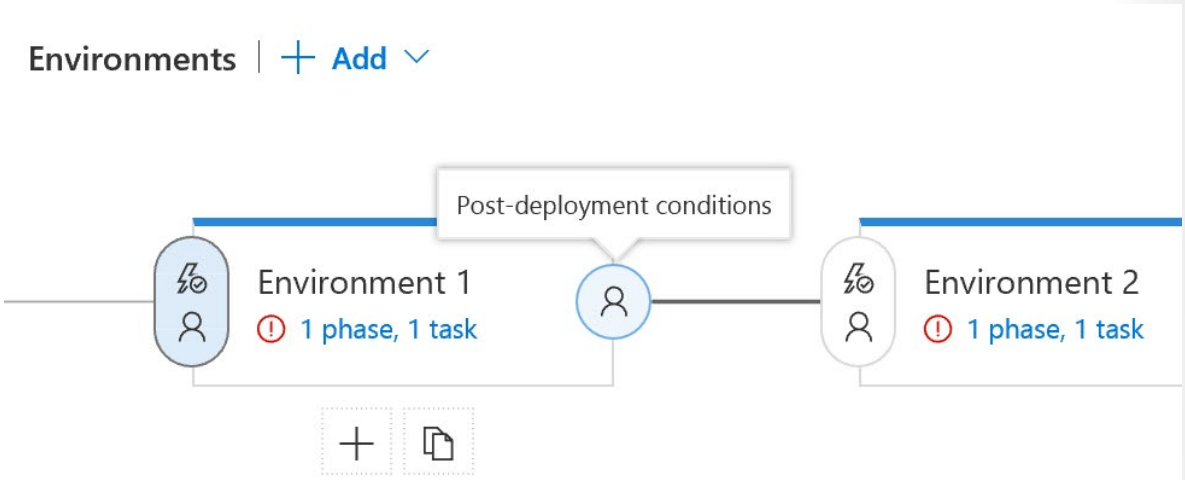
However, one of the things you'll quickly learn is that no amount of monitoring will catch every single problem and, particularly, if you have a popular application, your users will know within seconds and turn very quickly to Twitter to start asking about the problem. Twitter can be a wonderful "alert" to let you know something is wrong with your app.

The Twitter sentiment release gate that can be downloaded from the **Visual Studio Marketplace**[5] enables exactly this. It leverages Azure DevOps, Azure functions and Microsoft AI to analyze sentiment on your Twitter handle and gate your release progress based on it. The current implementation of the analysis is relatively simple and serves as a sample as much as anything else. It shows how easy it is to extend Azure DevOps release gates to measure any signal you choose and use that signal to manage your release process.

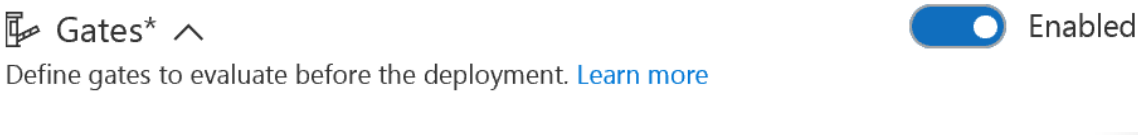
Once you install the Twitter sentiment extension from the marketplace, you'll need to follow the instructions to configure an Azure function to measure and analyze your sentiment. Then you can go into your Azure DevOps release pipelines and you will find a new release gate enabled.

Start by clicking on Post-deployment conditions on one of your environments.

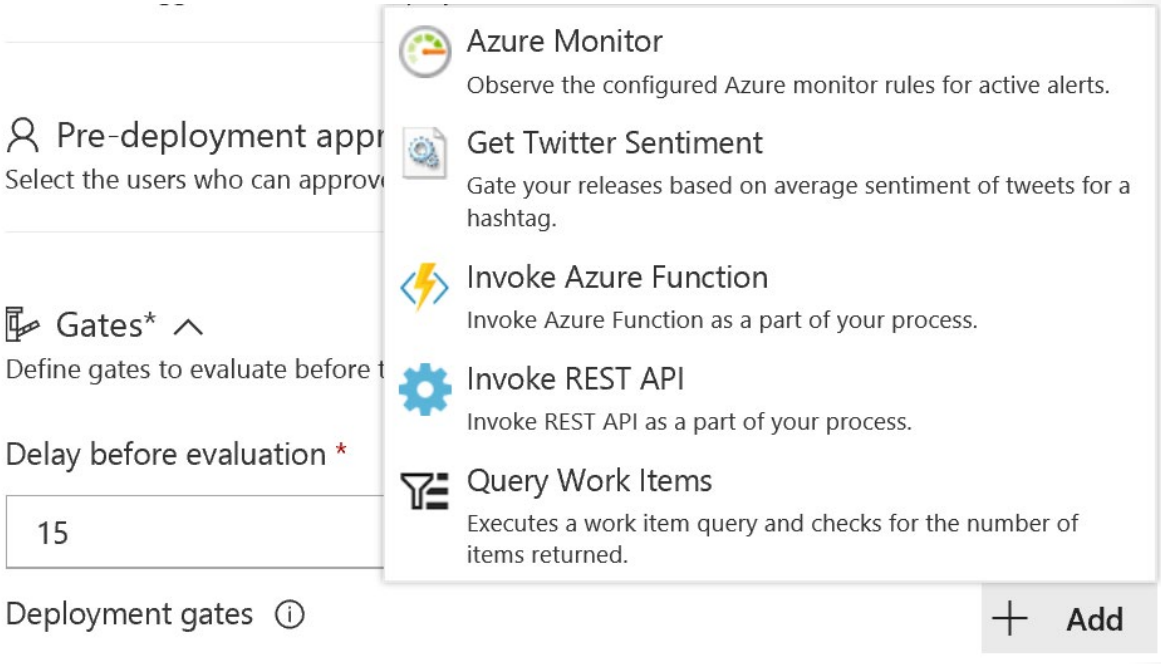
5 https://marketplace.visualstudio.com/items?itemName=ms-devlabs.vss-services-twittersentimentanalysis



Then enable the release gates.



Then choose the Twitter Sentiment item and configure it.



You can read up more about release gates **here**[6]

6 https://docs.microsoft.com/en-us/azure/devops/pipelines/release/approvals/gates?view=vsts

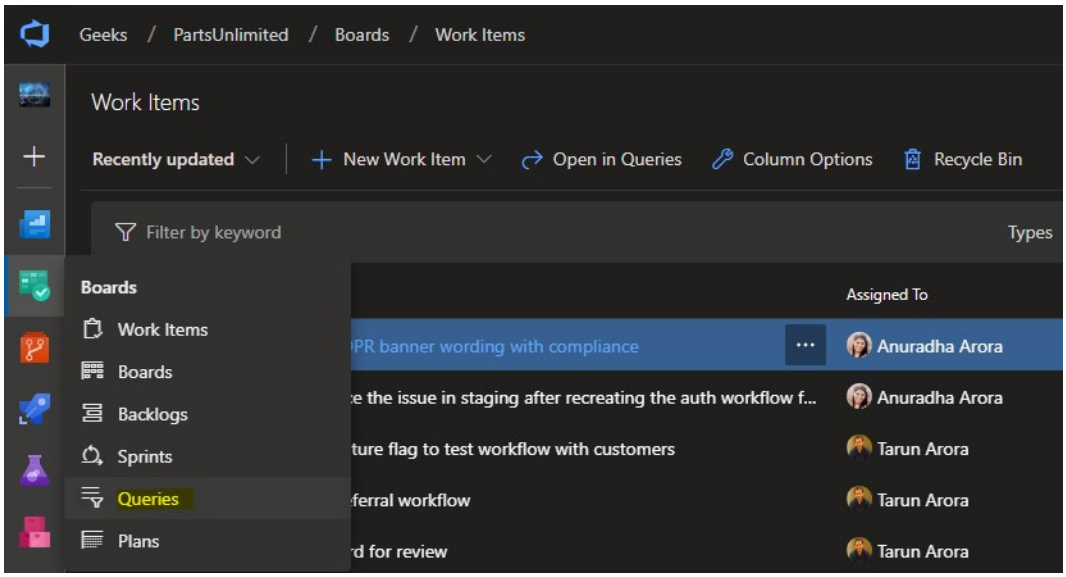
Deploy with RM Green light capabilities

A release process comprises of more than just deployment. Therefore a release pipeline needs more than just steps for application deployment. There are a whole raft of checks one would prefer to do before releasing a new version of the software through a release pipeline. To name a few... checks to ensure that there are no active critical issues open and being investigated, that there are no known service degradation or performance alerts. While the deployment process is automated, a lot of these checks are carried out manually which force automated release pipelines to have manual approval steps. No more! Azure Release Pipelines supports Gates.

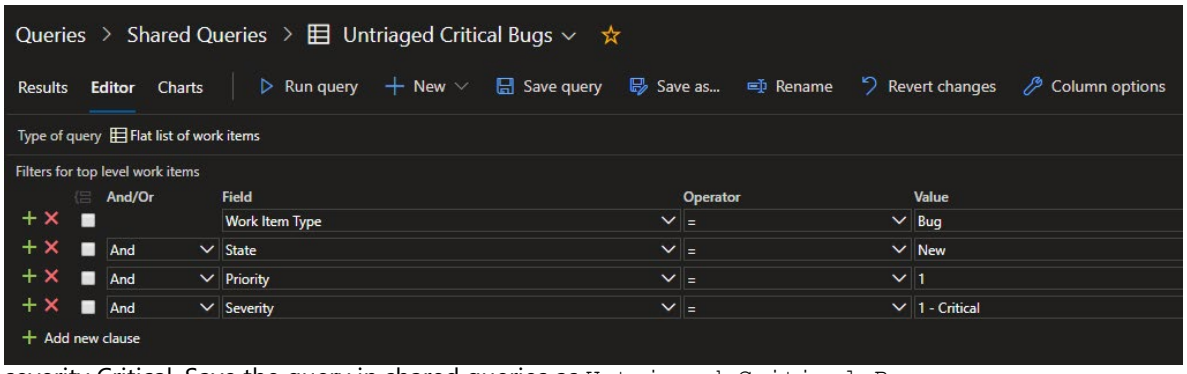
Gates allow automatic collection of health signals from external services, and then promote the release when all the signals are successful at the same time or stop the deployment on timeout. Typically, gates are used in connection with incident management, problem management, change management, monitoring, and external approval systems. Gates provide you a way to enable progressive exposure and phased deployments

Getting Started

1. Open the desired Team Project in AzureDevOps, navigate to the Queries page under Azure Boards.

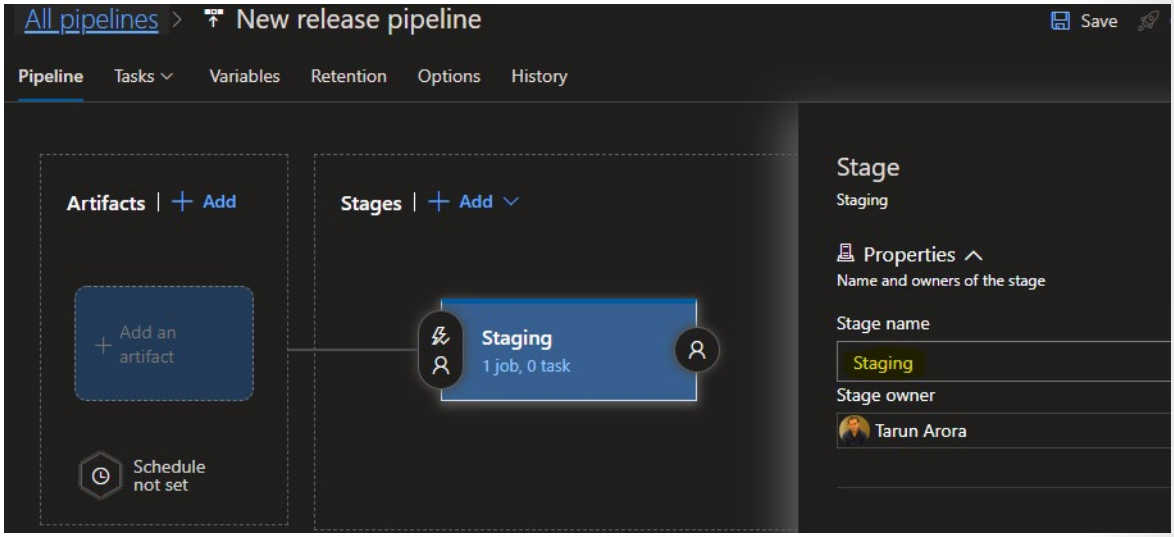


2. Click to create a new Query, add filter to return work item types Bug of status New with priority 1 and

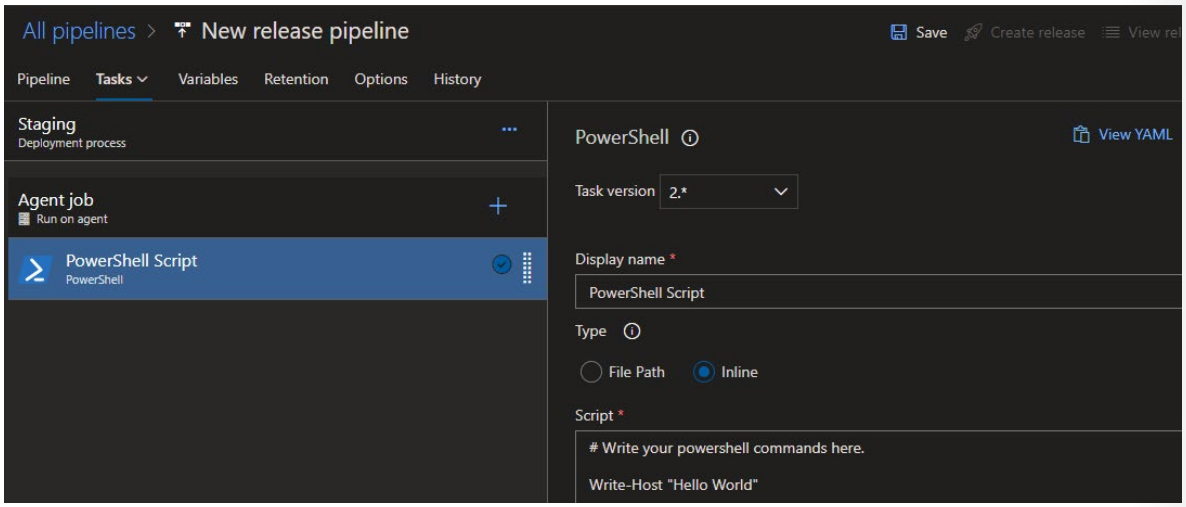


severity Critical. Save the query in shared queries as `Untriaged Critical Bugs`

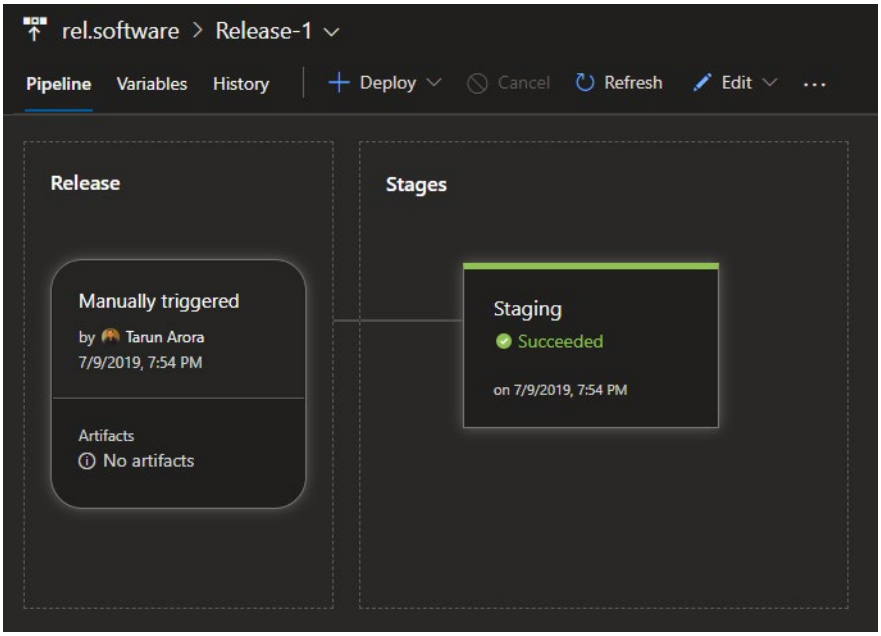
3. In the same team project, navigate to Azure Pipelines. Create an empty release pipeline with one stage, call it staging.



4. In Staging add a PowerShell task, set it to inline mode to print "Hello World"

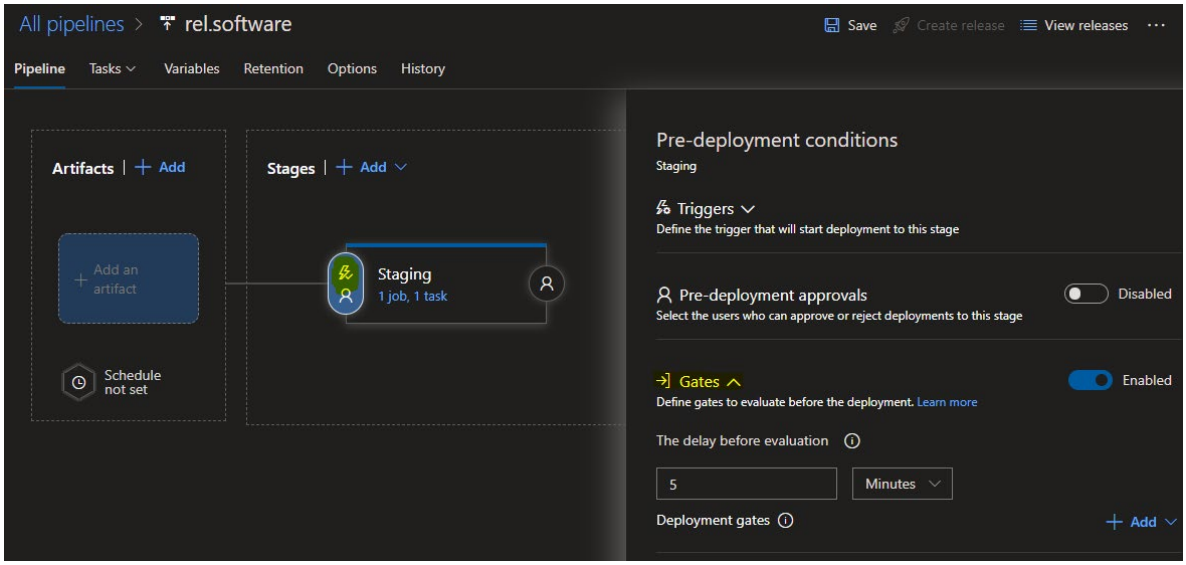


1. Name the pipeline as rel.software and click save. Create a release and see the release go through successfully.

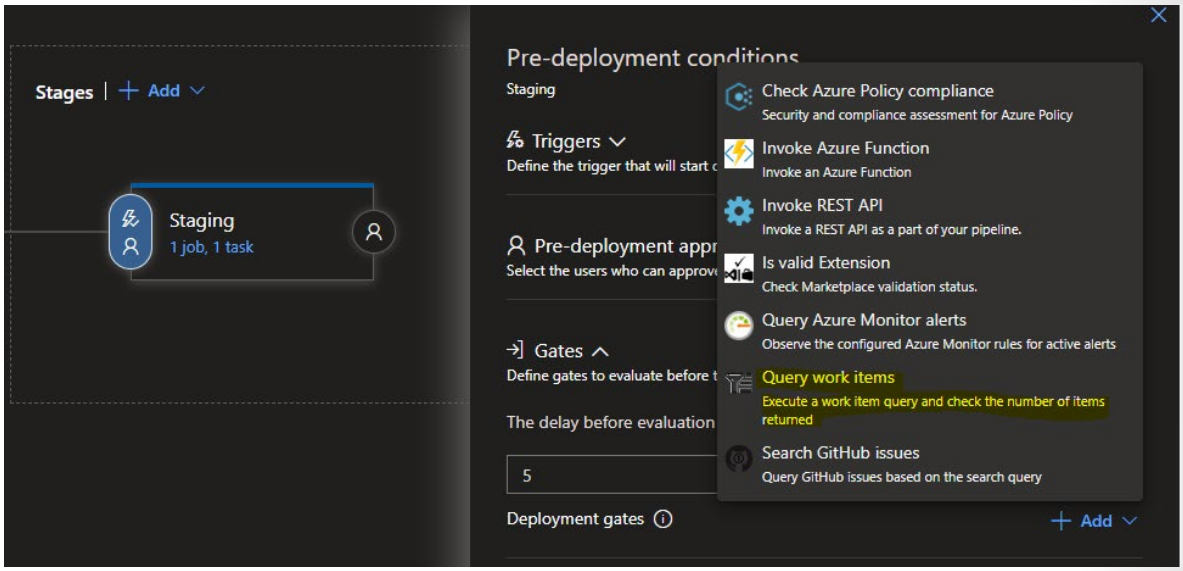


How to do it

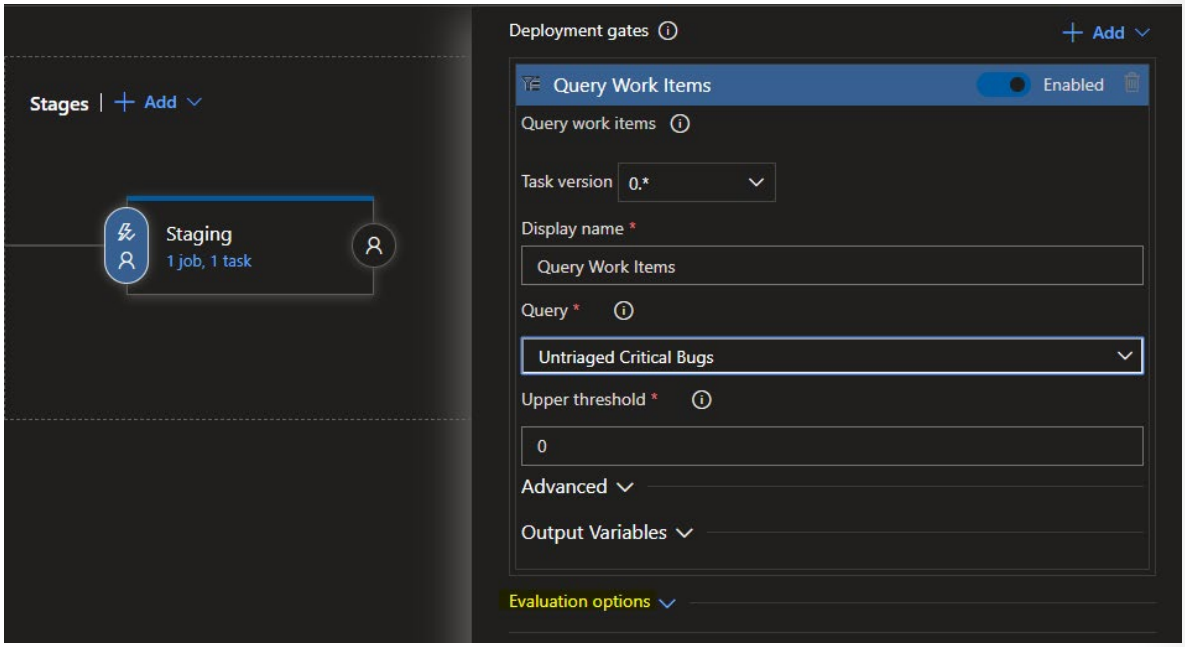
1. Edit the release pipeline, click on pre-deployment condition and from the right pane enable gates.



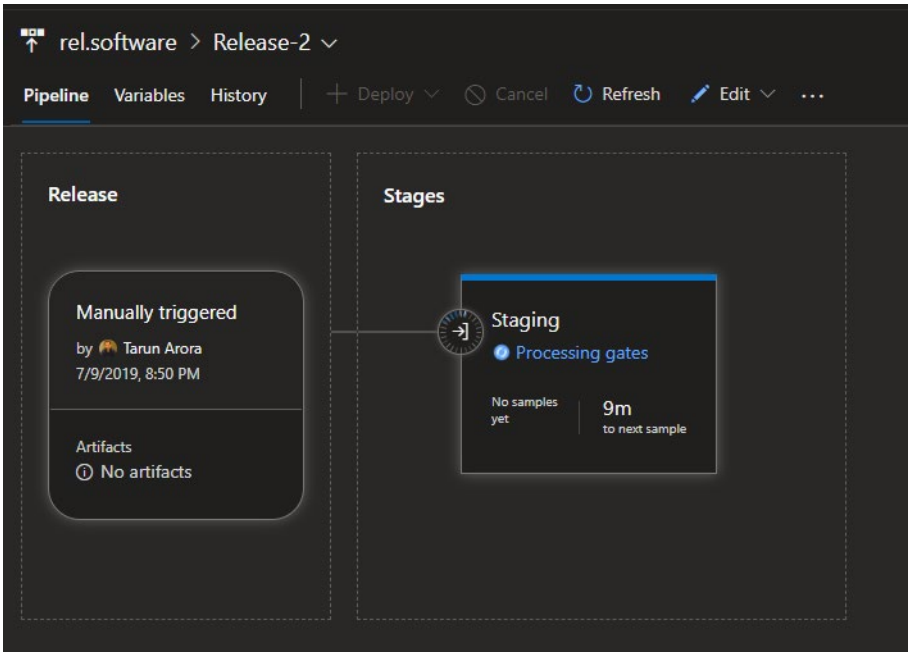
2. Click on Add to add a new gate, choose `Query Work Item` type Gate



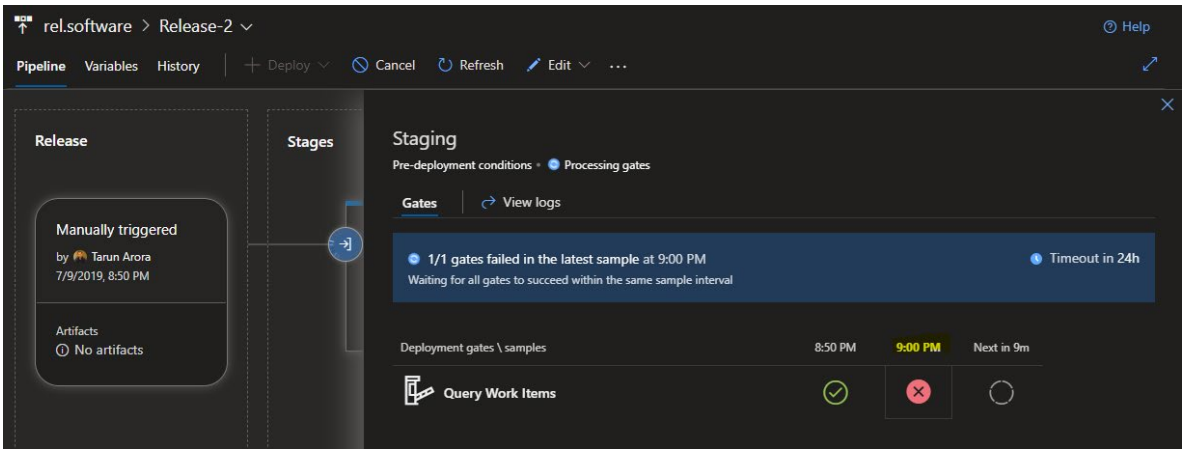
3. From the query work item drop down select the shared query 'untriaged critical bugs' created earlier and set the upper threshold to 0. Set the delay between evaluation period to 1 minute. Expand the evaluation options section and ensure that the time between evaluation of gates is set to 10 minutes and the Minimum duration for steady results results after a successful gates evaluation is configured to 15 minutes.



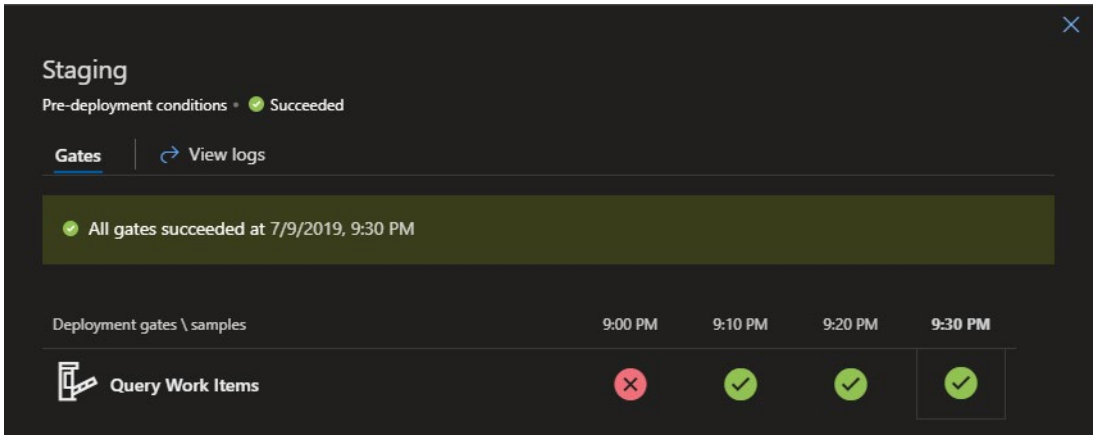
1. Save and queue a release. You'll see that the gate gets evaluated right away...



1. Now before the next evaluation of the gate, create a new work item of type Bug with Priority 1 and severity critical. Wait for the next evaluation of the gate in the release pipeline to complete.

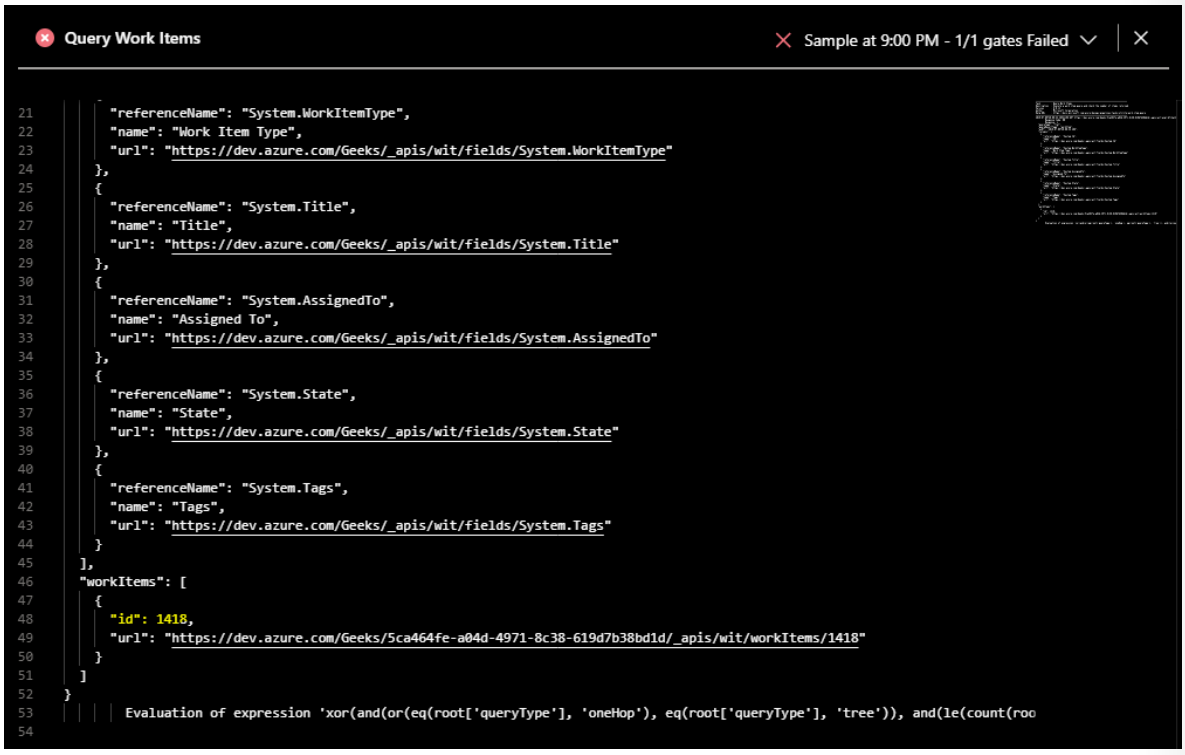


1. Close the bug as fixed, you'll see that after periodic evaluations and a stable period of 15 minutes the release is completed successfully.



How it works

1. When a new release is triggered, the release goes into a state of pre-approvals. At this time, the automated gate is evaluated at the specified interval of 10 minutes. The release will only move into approval state if the gate passes for the duration of steady results, specified as 15 minutes in this case. As you can can see in the logs below, the gate is failed in the 2nd validation check as one critical bug of 1 priority is identified in new state using the configured work item query.



1. Detailed logs of the gate checks can be downloaded and inspected along with the release pipeline logs.

There's more

In this tutorial we looked at the Work Item Query gate. The following other gates are supported out of the box.

The following gates are available by default:

- Invoke Azure function: Trigger execution of an Azure function and ensure a successful completion. For more details, see Azure function task.
- Query Azure monitor alerts: Observe the configured Azure monitor alert rules for active alerts. For more details, see Azure monitor task.
- Invoke REST API: Make a call to a REST API and continue if it returns a successful response. For more details, see HTTP REST API task.
- Query Work items: Ensure the number of matching work items returned from a query is within a threshold. For more details, see Work item query task.
- Security and compliance assessment: Assess Azure Policy compliance on resources within the scope of a given subscription and resource group, and optionally at a specific resource level. For more details, see Security Compliance and Assessment task.

In addition to this you can develop your own gates using the Azure DevOps API's, check out the gates developed by the community for inspiration **here**[7]

7 https://www.visualstudiogeeks.com/DevOps/IntegratingServiceNowWithVstsReleaseManagementUsingDeploymentGate

Design process to automate application analytics

Introduction

In an Agile environment, you may typically find multiple development teams that work simultaneously, introducing new code or code changes on a daily basis, and sometimes several times a day. In such a rapid environment it is extremely common to find problems that have "slipped through the cracks" to find themselves in the production environment. When these issues arise they have probably already impacted end-users, requiring a speedy resolution.

This means that teams have to conduct an extremely rapid investigation to identify the root cause of the problem. Identifying where these symptoms are coming from and then isolating the root cause is a challenging task. Symptoms can be found across various layers of a large hybrid IT environment, such as different servers/VMs, storage devices, databases, to the front-end and server side code. Investigations that traditionally would take hours or days to complete must be completed within minutes.

Teams must examine the infrastructure and application logs as part of this investigation process. However, the massive amount of log records that are produced in these environments makes it virtually impossible to do this manually. It's much like trying to find a needle in a haystack. In most cases these investigations are conducted through the use of log management and analysis systems that collect and aggregate these logs (from infrastructure elements and applications), centralizing them in a single place and then providing search capabilities to explore the data. These solutions make it possible to conduct the investigation, but they still rely completely on the investigation skills and the knowledge of the user. The user must know exactly what to search for, and have a deep understanding of the environment in order to use them effectively.

It's important to understand that the log files of applications are far less predictable than the log files of infrastructure elements. The errors are essentially messages and error numbers that have been introduced to the code by developers in a non-consistent manner. Consequently, in most cases search queries yield thousands of results and do not include important ones, even when the user is skilled. That leaves the user with the same "needle in the haystack" situation.

As they say, "the best place to hide a dead body is page 2 of Google's search results", and this is not very different. This process fails in light of the fast-paced reality of the DevOps activity.

Assisting DevOps with Augmented Search

A new breed of log management and analysis technologies has evolved to solve this challenge. These technologies expedite the identification and investigation processes through the use of Augmented Search. Designed specifically to deal with the chaotic and unpredictable nature of application logs, Augmented Search takes into consideration that users don't necessarily know what to search for, especially in the chaotic application layer environment.

The analysis algorithm automatically identifies errors, risk factors, and problem indicators, while analyzing their severity by combining semantic processing, statistical models, and machine learning to analyze and "understand" the events in the logs. These insights are displayed as intelligence layers on top of the search results, helping the user to quickly discover the most relevant and important information.

Although, DevOps engineers may be familiar with the infrastructure and system architecture the data is constantly changing with continuous fast pace deployment cycles and constant code changes. This means that DevOps teams can use their intuition and knowledge to start investigating each problem, but they have blind spots that consume time due to dynamic nature of the log data.

Combining the decisions that DevOps engineers makes during their investigation with the Augmented Search engine information layers on the important problems that occurred during the period of interest can help guide them through these blind spots quickly. The combination of the user's intellect, acquaintance with the system's architecture, and Augmented Search machine learning capabilities on the dynamic data makes it faster and easier to focus on the most relevant data. Here's how that works in practice:

One of the servers went down and any attempt to reinitiate the server has failed., However, since the process is running, the server seems to be up. In this case, end-users are complaining that an application is not responding. This symptom could be related to many problems in a complex environment with a large number of servers.

Focusing on the server that is behind this problem can be difficult, as it seems to be up. But finding the root cause of the problem requires a lengthy investigation even when you know which server is behind this problem.

Augmented Search will display a layer which highlights critical events that occurred during the specified time period instead of going over thousands of search results. These highlights provide information regarding the sources of the events, assisting in the triage process. At this point, DevOps enginners can understand the impact of the problem (e.g. which servers are affected by it) and then continue the investigation to find the root cause of these problems.

Using Augmented Search, DevOps engineers can identify a problem and the root cause in a matter of seconds instead of examining thousands of log events or running multiple checks on the various servers. Adding this type of visibility to log analysis, and the ability to surface critical events out of tens of thousands - and often millions - of events, is essential in a fast paced environment, in which changes are constantly introduced.

Recommend system and feature usage tracking tools

A key factor to automating feedback is telemetry. By inserting telemetric data into your production application and environment, the DevOps team can automate feedback mechanisms while monitoring applications in real-time. DevOps teams use telemetry to see and solve problems as they occur, but this data can be useful to both technical and business users.

When properly instrumented, telemetry can also be used to see and understand in real time how customers are engaging with the application. This could be critical information for product managers, marketing teams, and customer support. Thus it’s important that feedback mechanisms share continuous intelligence with all stakeholders.

What is Telemetry and Why Should I Care?

In the software development world, telemetry can offer insights on which features end users use most, detection of bugs and issues, and offering better visibility into performance without the need to solicit feedback directly from users. In DevOps and the world of modern cloud apps, we are tracking the health and performance of an application. That telemetry data comes from application logs, infrastructure logs, metrics and events. The measurements are things like memory consumption, CPU performance, and database response time, events can be used to measure everything else such as when a user logged in, when an item is added to a basket, when a sale is made, etc.

The concept of telemetry is often confused with just logging. But logging is a tool used in the development process to diagnose errors and code flows, and it’s focused on the internal structure of a website, app, or another development project. But logging only gives you a single dimension view, combined with insights of infrastructure logs, metrics and events you have a 360 degree view to understand user intend

and behaviour. Once a project is released, telemetry is what you're looking for to enable automatic collection of data from real-world use. Telemetry is what makes it possible to collect all that raw data that becomes valuable, actionable analytics.

Benefits of Telemetry

The primary benefit of telemetry is the ability of an end user to monitor the state of an object or environment while physically far removed from it. Once you've shipped a product, you can't be physically present, peering over the shoulders of thousands (or millions) of users as they engage with your product to find out what works, what's easy, and what's cumbersome. Thanks to telemetry, those insights can be delivered directly into a dashboard for you to analyze and act on.

Because telemetry provides insights into how well your product is working for your end users – as they use it – it's an incredibly valuable tool for ongoing performance monitoring and management. Plus, you can use the data you've gathered from version 1.0 to drive improvements and prioritize updates for your release of version 2.0.

Telemetry enables you to answer questions such as:

- Are your customers using the features you expect? How are they engaging with your product?
- How frequently are users engaging with your app, and for what duration?
- What settings options to users select most? Do they prefer certain display types, input modalities, screen orientation, or other device configurations?
- What happens when crashes occur? Are crashes happening more frequently when certain features or functions are used? What's the context surrounding a crash?

Obviously, the answers to these and the many other questions that can be answered with telemetry are invaluable to the development process, enabling you to make continuous improvements and introduce new features that, to your end users, may seem as though you've been reading their minds – which you have been, thanks to telemetry.

Challenges of Telemetry

Telemetry is clearly a fantastic technology, but it's not without its challenges. The most prominent challenge – and a commonly occurring issue – is not with telemetry itself, but with your end users and their willingness to allow what some see as Big Brother-esque spying. In short, some users immediately turn it off when they notice it, meaning any data generated from their use of your product won't be gathered or reported.

That means the experience of those users won't be accounted for when it comes to planning your future roadmap, fixing bugs, or addressing other issues in your app. Although this isn't necessarily a problem by itself, the issue is that users who tend to disallow these types of technologies can tend to fall into the more tech-savvy portion of your user base. This can result in the dumbing-down of software. Other users, on the other hand, take no notice to telemetry happening behind the scenes or simply ignore it if they do.

It's a problem without a clear solution — and it doesn't negate the overall power of telemetry for driving development — but one to keep in mind as you analyze your data. Therefore when designing a strategy for how you consider the feedback from application telemetry it's important to account for users who don't participate in providing the telemetry.

Recommend-monitoring-tools-and-technologies

Continuous monitoring of applications in production environments is typically implemented with application performance management (APM) solutions that intelligently monitor, analyze and manage cloud, on-premise and hybrid applications and IT infrastructure. These APM solutions enable you to monitor your users' experience and improve the stability of your application infrastructure. It helps identify the root cause of issues quickly to proactively prevent outages and keep users satisfied.

With a DevOps approach, we are also seeing more customers broaden the scope of continuous monitoring into the staging, testing and even development environments. This is possible because development and test teams that are following a DevOps approach are striving to use production-like environments for testing as much as possible. By running APM solutions earlier in the life cycle, development teams get feedback in advance of how applications will eventually perform in production and can take corrective action much earlier. In addition, operations teams that now are advising the development teams get advance knowledge and experience to better prepare and tune the production environment, resulting in far more stable releases into production.

Applications are more business critical than ever. They must be always up, always fast and always improving. Embracing a DevOps approach will allow you to reduce your cycle times to hours instead of months, but you have to keep ensuring a great user experience! Continuous monitoring of your entire DevOps life cycle will ensure development and operations teams collaborate to optimize the user experience every step of the way, leaving more time for your next big innovation.

When shortlisting a monitoring tool, you should seek the following advanced features:

Synthetic Monitoring: Developers, testers and operations staff all need to ensure that their internet and intranet mobile applications and web applications are tested and operate successfully from different points of presence around the world.

Alert Management: Developers, testers and operations staff all need to send notifications via email, voice mail, text, mobile push notifications and Slack messages when specific situations or events occur in development, testing or production environments, to get the right people's attention and to manage their response.

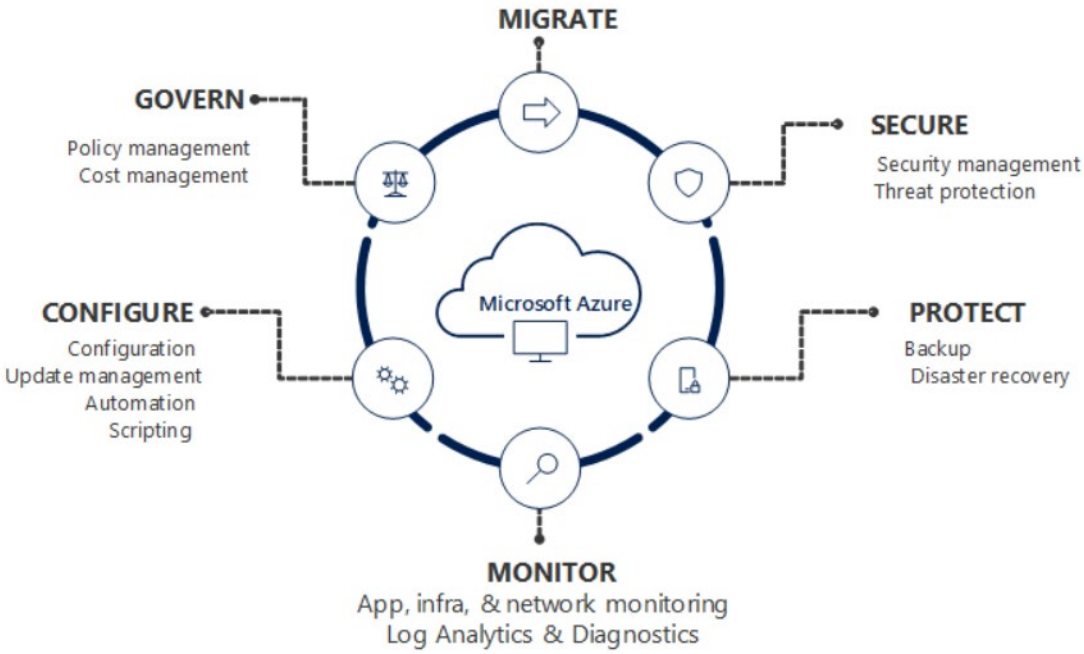
Deployment Automation: Developers, testers and operations staff use different tools to schedule and deploy complex applications and configure them in development, testing and production environments. We will discuss the best practices for these teams to collaborate effectively and efficiently and avoid potential duplication and erroneous information.

Analytics: Developers need to be able to look for patterns in log messages to identify if there is a problem in the code. Operations need to do root cause analysis across multiple log files to identify the source of the problem in complex application and systems.

Module 2 Implement Process for Routing System Feedback to Development Teams

Implement Tools to Track System Usage, Feature Usage, and Flow

Introduction



Continuous monitoring refers to the process and technology required to incorporate monitoring across each phase of your DevOps and IT operations lifecycles. It helps to continuously ensure the health, performance, and reliability of your application and infrastructure as it moves from development to production. Continuous monitoring builds on the concepts of Continuous Integration and Continuous

Deployment (CI/CD) which help you develop and deliver software faster and more reliably to provide continuous value to your users.

Azure Monitor[1] is the unified monitoring solution in Azure that provides full-stack observability across applications and infrastructure in the cloud and on-premises. It works seamlessly with **Visual Studio and Visual Studio Code**[2] during development and test and integrates with **Azure DevOps**[3] for release management and work item management during deployment and operations. It even integrates across the ITSM and SIEM tools of your choice to help track issues and incidents within your existing IT processes.

This article describes specific steps for using Azure Monitor to enable continuous monitoring throughout your workflows. It includes links to other documentation that provides details on implementing different features.

Enable monitoring for all your applications[4]

In order to gain observability across your entire environment, you need to enable monitoring on all your web applications and services. This will allow you to easily visualize end-to-end transactions and connections across all the components.

- **Azure DevOps Projects**[5] give you a simplified experience with your existing code and Git repository, or choose from one of the sample applications to create a Continuous Integration (CI) and Continuous Delivery (CD) pipeline to Azure.
- **Continuous monitoring in your DevOps release pipeline**[6] allows you to gate or rollback your deployment based on monitoring data.
- **Status Monitor**[7] allows you to instrument a live .NET app on Windows with Azure Application Insights, without having to modify or redeploy your code.
- If you have access to the code for your application, then enable full monitoring with **Application Insights**[8] by installing the Azure Monitor Application Insights SDK for **.NET**[9], **Java**[10], **Node.js**[11], or **any other programming languages**[12]. This allows you to specify custom events, metrics, or page views that are relevant to your application and your business.

Enable monitoring for your entire infrastructure[13]

Applications are only as reliable as their underlying infrastructure. Having monitoring enabled across your entire infrastructure will help you achieve full observability and make it easier to discover a potential root

1 https://docs.microsoft.com/en-us/azure/azure-monitor/overview
2 https://visualstudio.microsoft.com/
3 https://docs.microsoft.com/en-us/azure/devops/user-guide/index
4 https://docs.microsoft.com/en-us/azure/azure-monitor/continuous-monitoring#enable-monitoring-for-all-your-applications
5 https://docs.microsoft.com/en-us/azure/devops-project/overview
6 https://docs.microsoft.com/en-us/azure/application-insights/app-insights-vsts-continuous-monitoring
7 https://docs.microsoft.com/en-us/azure/application-insights/app-insights-monitor-performance-live-website-now
8 https://docs.microsoft.com/en-us/azure/application-insights/app-insights-overview
9 https://docs.microsoft.com/en-us/azure/application-insights/quick-monitor-portal
10 https://docs.microsoft.com/en-us/azure/application-insights/app-insights-java-quick-start
11 https://docs.microsoft.com/en-us/azure/application-insights/app-insights-nodejs-quick-start
12 https://docs.microsoft.com/en-us/azure/application-insights/app-insights-platforms
13 https://docs.microsoft.com/en-us/azure/azure-monitor/continuous-monitoring#enable-monitoring-for-your-entire-infrastructure

cause when something fails. Azure Monitor helps you track the health and performance of your entire hybrid infrastructure including resources such as VMs, containers, storage, and network.

- You automatically get **platform metrics, activity logs and diagnostics logs**[14] from most of your Azure resources with no configuration.
- Enable deeper monitoring for VMs with **Azure Monitor for VMs**[15].
- Enable deeper monitoring for AKS clusters with **Azure Monitor for containers**[16].
- Add **monitoring solutions**[17] for different applications and services in your environment.

Infrastructure as code[18] is the management of infrastructure in a descriptive model, using the same versioning as DevOps teams use for source code. It adds reliability and scalability to your environment and allows you to leverage similar processes that used to manage your applications.

- Use **Resource Manager templates**[19] to enable monitoring and configure alerts over a large set of resources.
- Use **Azure Policy**[20] to enforce different rules over your resources. This ensures that those resources stay compliant with your corporate standards and service level agreements.

Combine resources in Azure Resource Groups[21]

A typical application on Azure today includes multiple resources such as VMs and App Services or microservices hosted on Cloud Services, AKS clusters, or Service Fabric. These applications frequently utilize dependencies like Event Hubs, Storage, SQL, and Service Bus.

- Combine resources inAzure Resource Groups to get full visibility across all your resources that make up your different applications. **Azure Monitor for Resource Groups**[22] provides a simple way to keep track of the health and performance of your entire full-stack application and enables drilling down into respective components for any investigations or debugging.

Ensure quality through Continuous Deployment[23]

Continuous Integration / Continuous Deployment allows you to automatically integrate and deploy code changes to your application based on the results of automated testing. It streamlines the deployment process and ensures the quality of any changes before they move into production.

- Use **Azure Pipelines**[24] to implement Continuous Deployment and automate your entire process from code commit to production based on your CI/CD tests.

14 https://docs.microsoft.com/en-us/azure/azure-monitor/platform/data-sources
15 https://docs.microsoft.com/en-us/azure/azure-monitor/insights/vminsights-overview
16 https://docs.microsoft.com/en-us/azure/azure-monitor/insights/container-insights-overview
17 https://docs.microsoft.com/en-us/azure/azure-monitor/insights/solutions-inventory
18 https://docs.microsoft.com/en-us/azure/devops/learn/what-is-infrastructure-as-code
19 https://docs.microsoft.com/en-us/azure/azure-monitor/platform/template-workspace-configuration
20 https://docs.microsoft.com/en-us/azure/governance/policy/overview
21 https://docs.microsoft.com/en-us/azure/azure-monitor/continuous-monitoring#combine-resources-in-azure-resource-groups
22 https://docs.microsoft.com/en-us/azure/azure-monitor/insights/resource-group-insights
23 https://docs.microsoft.com/en-us/azure/azure-monitor/continuous-monitoring#ensure-quality-through-continuous-deployment
24 https://docs.microsoft.com/en-us/azure/devops/pipelines

- Use Quality Gates to integrate monitoring into your pre-deployment or post-deployment. This ensures that you are meeting the key health/performance metrics (KPIs) as your applications move from dev to production and any differences in the infrastructure environment or scale is not negatively impacting your KPIs.
- **Maintain separate monitoring instances**[25] between your different deployment environments such as Dev, Test, Canary, and Prod. This ensures that collected data is relevant across the associated applications and infrastructure. If you need to correlate data across environments, you can use **multi-resource charts in Metrics Explorer**[26] or create **cross-resource queries in Log Analytics**[27].

Create actionable alerts with actions[28]

A critical aspect of monitoring is proactively notifying administrators of any current and predicted issues.

- Create **alerts in Azure Monitor**[29] based on logs and metrics to identify predictable failure states. You should have a goal of making all alerts actionable meaning that they represent actual critical conditions and seek to reduce false positives. Use **dynamic thresholds**[30] to automatically calculate baselines on metric data rather than defining your own static thresholds.
- Define actions for alerts to use the most effective means of notifying your administrators. Available **actions for notification**[31] are SMS, e-mails, push notifications, or voice calls.
- Use more advanced actions to **connect to your ITSM tool**[32] or other alert management systems through **webhooks**[33].
- Remediate situations identified in alerts as well with **Azure Automation runbooks**[34] or **Logic Apps**[35] that can be launched from an alert using webhooks.
- Use **autoscaling**[36] to dynamically increase and decrease your compute resources based on collected metrics.

25 https://docs.microsoft.com/en-us/azure/application-insights/app-insights-separate-resources
26 https://docs.microsoft.com/en-us/azure/azure-monitor/platform/metrics-charts
27 https://docs.microsoft.com/en-us/azure/azure-monitor/log-query/cross-workspace-query
28 https://docs.microsoft.com/en-us/azure/azure-monitor/continuous-monitoring#create-actionable-alerts-with-actions
29 https://docs.microsoft.com/en-us/azure/azure-monitor/platform/alerts-overview
30 https://docs.microsoft.com/en-us/azure/azure-monitor/platform/alerts-dynamic-thresholds
31 https://docs.microsoft.com/en-us/azure/azure-monitor/platform/action-groups#create-an-action-group-by-using-the-azure-portal
32 https://docs.microsoft.com/en-us/azure/azure-monitor/platform/itsmc-overview
33 https://docs.microsoft.com/en-us/azure/azure-monitor/platform/activity-log-alerts-webhook
34 https://docs.microsoft.com/en-us/azure/automation/automation-webhooks
35 https://docs.microsoft.com/en-us/connectors/custom-connectors/create-webhook-trigger
36 https://docs.microsoft.com/en-us/azure/azure-monitor/learn/tutorial-autoscale-performance-schedule

Prepare dashboards and workbooks[37]

Ensuring that your development and operations have access to the same telemetry and tools allows them to view patterns across your entire environment and minimize your Mean Time To Detect (MTTD) and Mean Time To Restore (MTTR).

- Prepare **custom dashboards**[38] based on common metrics and logs for the different roles in your organization. Dashboards can combine data from all Azure resources.
- Prepare **Workbooks**[39] to ensure knowledge sharing between development and operations. These could be prepared as dynamic reports with metric charts and log queries, or even as troubleshooting guides prepared by developers helping customer support or operations to handle basic problems.

Continuously optimize[40]

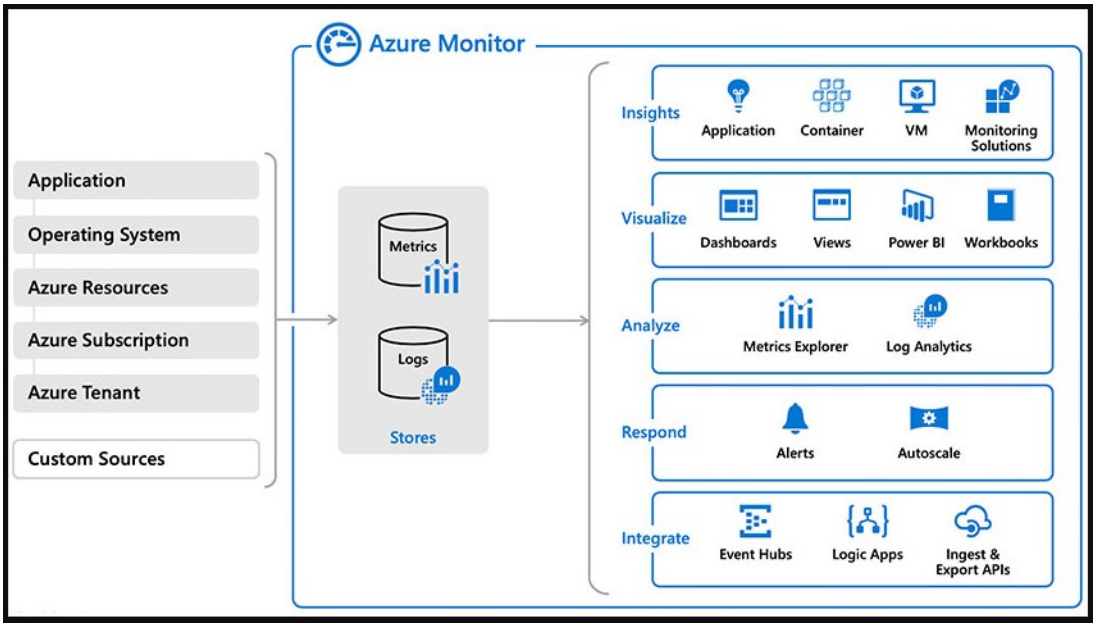
Monitoring is one of the fundamental aspects of the popular Build-Measure-Learn philosophy, which recommends continuously tracking your KPIs and user behavior metrics and then striving to optimize them through planning iterations. Azure Monitor helps you collect metrics and logs relevant to your business and to add new data points in the next deployment as required.

- Use tools in Application Insights to **track end-user behavior and engagement**[41].
- Use **Impact Analysis**[42] to help you prioritize which areas to focus on to drive to important KPIs.

Azure Log Analytics

When you run at cloud scale, you need intelligent logging and monitoring tools that scale to your needs and provide insight on your data in real time. Azure Monitor is Microsoft's native cloud monitoring solution, Azure Monitor collects monitoring telemetry from a variety of on-premises and Azure sources. Azure Monitor provides Management tools, such as those in Azure Security Center and Azure Automation, also enables ingestion of custom log data to Azure. The service aggregates and stores this telemetry in a log data store that's optimised for cost and performance. With Azure Monitor you can analyse data, set up alerts, get end-to-end views of your applications, and use machine learning-driven insights to quickly identify and resolve problems.

37 https://docs.microsoft.com/en-us/azure/azure-monitor/continuous-monitoring#prepare-dashboards-and-workbooks
38 https://docs.microsoft.com/en-us/azure/application-insights/app-insights-tutorial-dashboards
39 https://docs.microsoft.com/en-us/azure/application-insights/app-insights-usage-workbooks
40 https://docs.microsoft.com/en-us/azure/azure-monitor/continuous-monitoring#continuously-optimize
41 https://docs.microsoft.com/en-us/azure/application-insights/app-insights-tutorial-users
42 https://docs.microsoft.com/en-us/azure/application-insights/app-insights-usage-impact



In this tutorial we'll focus on the Log Analytics part of Azure Monitor. We'll learn how to,

- Set up Log Analytics workspace
- Connect virtual machines into a long analytics workspace
- Configure Log Analytics workspace to collect custom performance counters
- Analyse the telemetry using Kusto Query Language

Getting Started

1. To follow along you'll need a resource group with one or more virtual machines that you have RDP access to.
2. Log into **Azure Shell**[43]. Executing the command below will create a new resource group and create a new log analytics workspace. Take a note of the workspaceid of the log analytics workspace as we'll be using it again.

```
$ResourceGroup = "azwe-rg-devtest-logs-001"
$WorkspaceName = "azwe-devtest-logs-01"
$Location = "westeurope"

# List of solutions to enable
$Solutions = "CapacityPerformance", "LogManagement", "ChangeTracking",
"ProcessInvestigator"

# Create the resource group if needed
try {
    Get-AzResourceGroup -Name $ResourceGroup -ErrorAction Stop
} catch {
    New-AzResourceGroup -Name $ResourceGroup -Location $Location
}
```

43 http://shell.azure.com/powershell

```
# Create the workspace
New-AzOperationalInsightsWorkspace -Location $Location -Name $WorkspaceName
-Sku Standard -ResourceGroupName $ResourceGroup

# List all solutions and their installation status
Get-AzOperationalInsightsIntelligencePacks -ResourceGroupName $Resource-
Group -WorkspaceName $WorkspaceName

# Add solutions
foreach ($solution in $Solutions) {
    Set-AzOperationalInsightsIntelligencePack -ResourceGroupName $Resource-
Group -WorkspaceName $WorkspaceName -IntelligencePackName $solution -Ena-
bled $true
}

# List enabled solutions
(Get-AzOperationalInsightsIntelligencePacks -ResourceGroupName $Resource-
Group -WorkspaceName $WorkspaceName).Where({($_.enabled -eq $true)})

# Enable IIS Log Collection using agent
Enable-AzOperationalInsightsIISLogCollection -ResourceGroupName $Resource-
Group -WorkspaceName $WorkspaceName

# Windows Event
New-AzOperationalInsightsWindowsEventDataSource -ResourceGroupName $Re-
sourceGroup -WorkspaceName $WorkspaceName -EventLogName "Application"
-CollectErrors -CollectWarnings -Name "Example Application Event Log"
```

1. Retrieve the Log Analytics workspace secure key

```
Get-AzOperationalInsightsWorkspaceSharedKey `
                    -ResourceGroupName azwe-rg-devtest-logs-001 `
                    -Name azwe-devtest-logs-01
```

1. Map existing virtual machines with the Log Analytics workspace. The query below uses the wokspaceid and workspace secret key of the log analytics workspace to install the Microsoft Enterprise Cloud Monitoring extension onto an existing VM.

```
$PublicSettings = @{"workspaceId" = "<myWorkspaceId>"}
$ProtectedSettings = @{"workspaceKey" = "<myWorkspaceKey>"}

Set-AzVMExtension -ExtensionName "Microsoft.EnterpriseCloud.Monitoring" `
    -ResourceGroupName "azwe-rg-devtest-logs-001" `
    -VMName "azsu-d-sql01-01" `
    -Publisher "Microsoft.EnterpriseCloud.Monitoring" `
    -ExtensionType "MicrosoftMonitoringAgent" `
    -TypeHandlerVersion 1.0 `
    -Settings $PublicSettings `
```

```
    -ProtectedSettings $ProtectedSettings `
    -Location westeurope
```

1. Run the script to configure the below listed performance counters to be collected from the virtual machine

```
#Login-AzureRmAccount

#Instance
###################################
$InstanceNameAll = "*"
$InstanceNameTotal = '_Total'
#Objects
###################################
$ObjectCache = "Cache"
$ObjectLogicalDisk = "LogicalDisk"
$ObjectMemory = "Memory"
$ObjectNetworkAdapter = "Network Adapter"
$ObjectNetworkInterface = "Network Interface"
$ObjectPagingFile = "Paging File"
$ObjectProcess = "Process"
$ObjectProcessorInformation = "Processor Information"
$ObjectProcessor = "Processor"

$ObjectSQLAgentAlerts = "SQLAgent:Alerts"
$ObjectSQLAgentJobs = "SQLAgent:Jobs"
$ObjectSQLAgentStatistics = "SQLAgent:Statistics"

$ObjectSQLServerAccessMethods = "SQLServer:Access Methods"
$ObjectSQLServerExecStatistics = "SQLServer:Exec Statistics"
$ObjectSQLServerLocks = "SQLServer:Locks"
$ObjectSQLServerSQLErrors = "SQLServer:SQL Errors"

$ObjectSystem = "System"

#Counters
###################################################################
$CounterCache = "Copy Read Hits %"

$CounterLogicalDisk =
     "% Free Space" `
    ,"Avg. Disk sec/Read" `
    ,"Avg. Disk sec/Transfer" `
    ,"Avg. Disk sec/Write" `
    ,"Current Disk Queue Length" `
    ,"Disk Read Bytes/sec" `
    ,"Disk Reads/sec" `
    ,"Disk Transfers/sec" `
    ,"Disk Writes/sec"
```

```
$CounterMemory =
     "% Committed Bytes In Use" `
    ,"Available MBytes" `
    ,"Page Faults/sec" `
    ,"Pages Input/sec" `
    ,"Pages Output/sec" `
    ,"Pool Nonpaged Bytes"

$CounterNetworkAdapter =
     "Bytes Received/sec" `
    ,"Bytes Sent/sec"

$CounterNetworkInterface = "Bytes Total/sec"

$CounterPagingFile =
     "% Usage" `
    ,"% Usage Peak"

$CounterProcess = "% Processor Time"

$CounterProcessorInformation =
     "% Interrupt Time" `
    ,"Interrupts/sec"

$CounterProcessor = "% Processor Time"
$CounterProcessorTotal = "% Processor Time"

$CounterSQLAgentAlerts = "Activated alerts"
$CounterSQLAgentJobs = "Failed jobs"
$CounterSQLAgentStatistics = "SQL Server restarted"
$CounterSQLServerAccessMethods = "Table Lock Escalations/sec"
$CounterSQLServerExecStatistics = "Distributed Query"
$CounterSQLServerLocks = "Number of Deadlocks/sec"
$CounterSQLServerSQLErrors = "Errors/sec"

$CounterSystem = "Processor Queue Length"

############################################################
$global:number = 1 #Name parameter needs to be unique that why we will use
number ++ in fuction
############################################################

function AddPerfCounters ($PerfObject, $PerfCounters, $Instance)
{
    ForEach ($Counter in $PerfCounters)
    {
        New-AzureRmOperationalInsightsWindowsPerformanceCounterDataSource `
                    -ResourceGroupName 'azwe-rg-devtest-logs-001' `
                    -WorkspaceName 'azwe-devtest-logs-001' `
                    -ObjectName $PerfObject `
                    -InstanceName $Instance `
```

```
                    -CounterName $Counter `
                    -IntervalSeconds 10 `
                    -Name "Windows Performance Counter $global:number"
        $global:number ++
    }
}

AddPerfCounters -PerfObject $ObjectLogicalDisk -PerfCounter $CounterLogi-
calDisk -Instance $InstanceNameAll
AddPerfCounters -PerfObject $ObjectNetworkAdapter -PerfCounter $CounterNet-
workAdapter -Instance $InstanceNameAll
AddPerfCounters -PerfObject $ObjectNetworkInterface -PerfCounter $Counter-
NetworkInterface -Instance $InstanceNameAll
AddPerfCounters -PerfObject $ObjectPagingFile -PerfCounter $CounterPaging-
File -Instance $InstanceNameAll
AddPerfCounters -PerfObject $ObjectProcess -PerfCounter $CounterProcess
-Instance $InstanceNameAll
AddPerfCounters -PerfObject $ObjectProcessorInformation -PerfCounter $Coun-
terProcessorInformation -Instance $InstanceNameAll
AddPerfCounters -PerfObject $ObjectProcessor -PerfCounter $CounterProcessor
-Instance $InstanceNameAll
AddPerfCounters -PerfObject $ObjectProcessor -PerfCounter $CounterProces-
sorTotal -Instance $InstanceNameTotal
AddPerfCounters -PerfObject $ObjectSQLAgentAlerts -PerfCounter $Counter-
SQLAgentAlerts -Instance $InstanceNameAll
AddPerfCounters -PerfObject $ObjectSQLAgentJobs -PerfCounter $CounterSQLA-
gentJobs -Instance $InstanceNameAll
AddPerfCounters -PerfObject $ObjectSQLAgentStatistics -PerfCounter $Coun-
terSQLAgentStatistics -Instance $InstanceNameAll
AddPerfCounters -PerfObject $ObjectSQLServerAccessMethods -PerfCounter
$CounterSQLServerAccessMethods -Instance $InstanceNameAll
AddPerfCounters -PerfObject $ObjectSQLServerExecStatistics -PerfCounter
$CounterSQLServerExecStatistics -Instance $InstanceNameAll
AddPerfCounters -PerfObject $ObjectSQLServerLocks -PerfCounter $CounterSQL-
ServerLocks -Instance $InstanceNameAll
AddPerfCounters -PerfObject $ObjectSQLServerSQLErrors -PerfCounter $Coun-
terSQLServerSQLErrors -Instance $InstanceNameAll
AddPerfCounters -PerfObject $ObjectSystem -PerfCounter $CounterSystem
-Instance $InstanceNameAll
AddPerfCounters -PerfObject $ObjectMemory -PerfCounter $CounterMemory
-Instance $InstanceNameAll
AddPerfCounters -PerfObject $ObjectCache -PerfCounter $CounterCache -In-
stance $InstanceNameAll
```

1. In order to generate some interesting performance statistics. Download **HeavyLoad utility**[44] (a free load testing utility) and run this on the virtual machine to simulate high CPU, Memory and IOPS consumption.

44 https://www.jam-software.com/heavyload/

Summary - So far, we've created a log analytics workspace in a resource group. The log analytics workspace has been configured to collect performance counters, event logs and IIS Logs. A virtual machine has been mapped to the log analytics workspace using the Microsoft Enterprise cloud monitoring extension. HeavyLoad has been used to simulate high CPU, memory and IOPS on the virtual machine.

How to

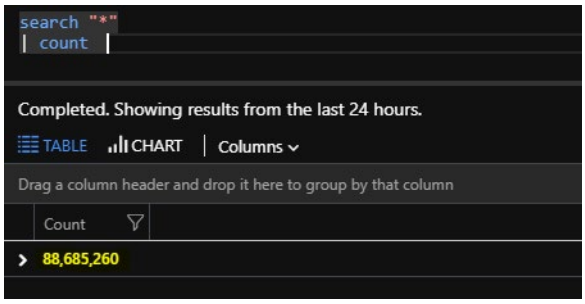
1. Log into **Azure Portal**[45] and navigate to the log analytics workspace. From the left blade in the log analytics workspace click Logs. This will open up the Logs window, ready for you to start exploring all the datapoints captured into the workspace.
2. To query the logs we'll need to use the Kusto Query Language. Run the query below to list the last heartbeat of each machine connected to the log analytics workspace

```
// Last heartbeat of each computer
// Show the last heartbeat sent by each computer
Heartbeat
| summarize arg_max(TimeGenerated, *) by Computer
```

1. Show a list of all distinct counters being captured

```
// What data is being collected?
// List the collected performance counters and object types (Process,
Memory, Processor..)
Perf
| summarize by ObjectName, CounterName
```

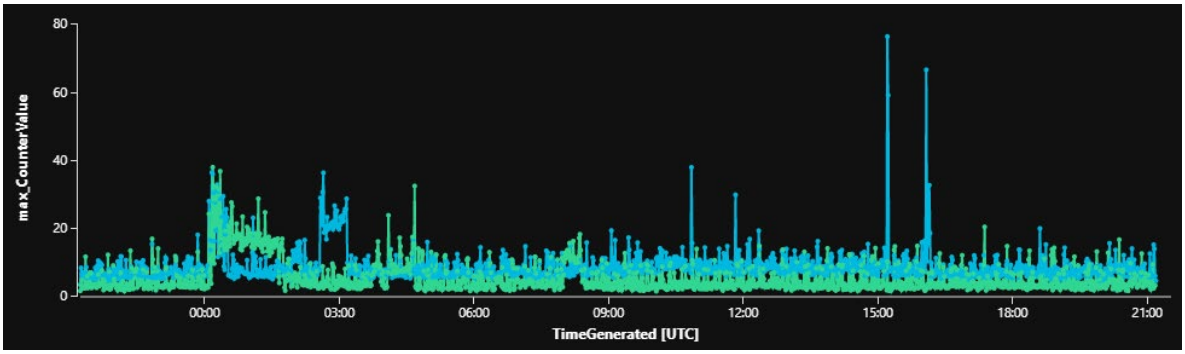
1. Show a count of the data points collected in the last 24 hours. In the result below, you can see we have 66M data points that we are able to query against in near real time to analyse and correlate insights



1. Run the query below to generate the max CPU Utilization trend over the last 24 hours, aggregated at a granularity of 1 min. Render the data as timechart.

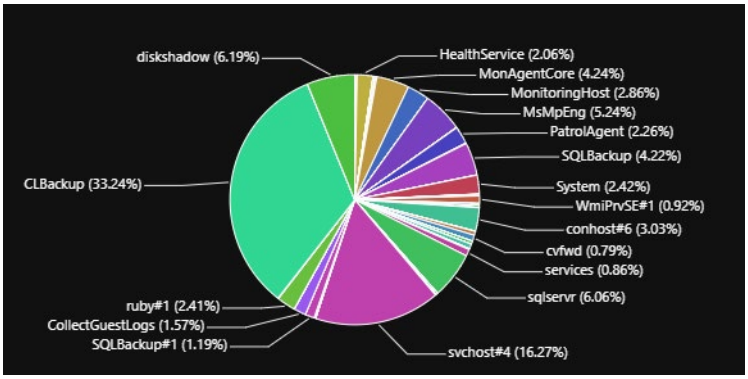
```
Perf
| where ObjectName == "Processor" and InstanceName == "_Total"
| summarize max(CounterValue) by Computer, bin(TimeGenerated, 1m)
| render  timechart
```

45 https://portal.azure.com



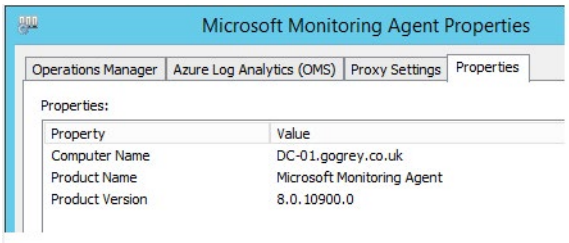
1. Run the query below to see all the processes running on that machine that are contributing to the CPU Utilization. Render the data in a piechart.

```
Perf
| where ObjectName contains "process"
        and InstanceName !in ("_Total", "Idle")
        and CounterName == "% Processor Time"
| summarize avg(CounterValue) by InstanceName, CounterName, bin(TimeGener-
ated, 1m)
| render piechart
```



How it works

1. Log Analytics works by running the Microsoft Monitoring Agent service on the machine. The service locally captures and buffers the events and pushes them securely out to the Log Analytics workspace in Azure.
2. Log into the virtual machine and navigate to the C:\Program Files\Microsoft Monitoring Agent\MMA and open control panel. This will show you the details of the log analytics workspace connected. You also have the option of adding multiple log analytics workspaces to publish the log data into multiple workspaces.



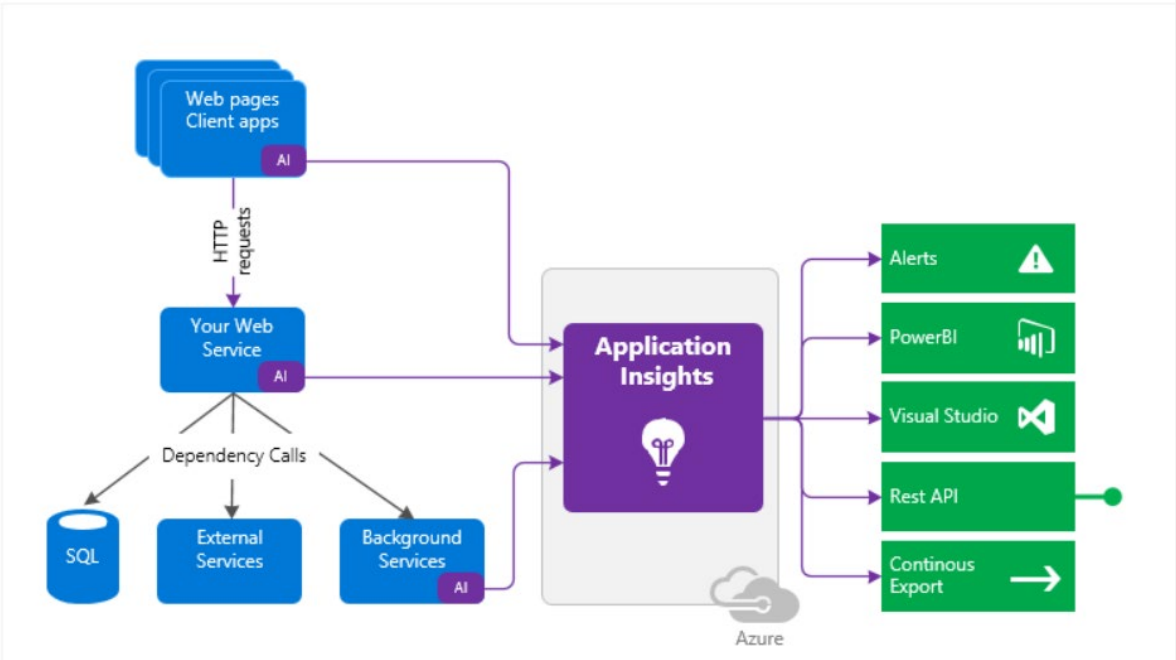
There's more

This tutorial has introduced you to the basic concepts of Log Analytics and how to get started with the basics. We've only scratched the surface of what's possible with Log Analytics. I would encourage you to try out the advanced tutorials available for Log Analytics on **Microsoft Docs**[46]

How Does Application Insights Work

You install a small instrumentation package in your application, and set up an Application Insights resource in the Microsoft Azure portal. The instrumentation monitors your app and sends telemetry data to the portal. (The application can run anywhere - it doesn't have to be hosted in Azure.)

You can instrument not only the web service application, but also any background components, and the JavaScript in the web pages themselves.



In addition, you can pull in telemetry from the host environments such as performance counters, Azure diagnostics, or Docker logs. You can also set up web tests that periodically send synthetic requests to your web service.

46 https://docs.microsoft.com/en-us/azure/azure-monitor/

All these telemetry streams are integrated in the Azure portal, where you can apply powerful analytic and search tools to the raw data.

What's the overhead?[47]

The impact on your app's performance is very small. Tracking calls are non-blocking, and are batched and sent in a separate thread.

What does Application Insights monitor?[48]

Application Insights is aimed at the development team, to help you understand how your app is performing and how it's being used. It monitors:

- Request rates, response times, and failure rates - Find out which pages are most popular, at what times of day, and where your users are. See which pages perform best. If your response times and failure rates go high when there are more requests, then perhaps you have a resourcing problem.
- Dependency rates, response times, and failure rates - Find out whether external services are slowing you down.
- Exceptions - Analyse the aggregated statistics, or pick specific instances and drill into the stack trace and related requests. Both server and browser exceptions are reported.
- Page views and load performance - reported by your users' browsers.
- AJAX calls from web pages - rates, response times, and failure rates.
- User and session counts.
- Performance counters from your Windows or Linux server machines, such as CPU, memory, and network usage.
- Host diagnostics from Docker or Azure.
- Diagnostic trace logs from your app - so that you can correlate trace events with requests.
- Custom events and metrics that you write yourself in the client or server code, to track business events such as items sold or games won.

Where do I see my telemetry?

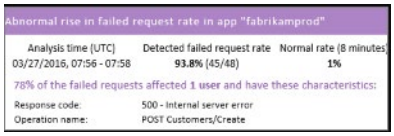
There are plenty of ways to explore your data. Check out these articles:

47 https://docs.microsoft.com/en-us/azure/application-insights/app-insights-overview?toc=/azure/azure-monitor/toc.json#whats-the-overhead

48 https://docs.microsoft.com/en-us/azure/application-insights/app-insights-overview?toc=/azure/azure-monitor/toc.json#what-does-application-insights-monitor

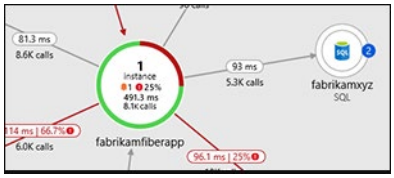
Smart detection and manual alerts[49]

Automatic alerts adapt to your app's normal patterns of telemetry and trigger when there's something outside the usual pattern. You can also **set alerts**[50] on particular levels of custom or standard metrics.



Application map[51]

The components of your app, with key metrics and alerts.



Profiler[52]

Inspect the execution profiles of sampled requests.

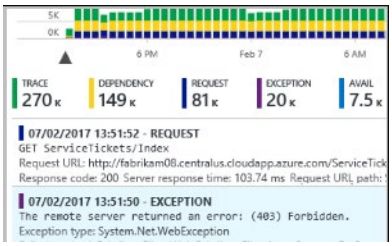
Usage analysis[53]

Analyze user segmentation and retention.

49 https://docs.microsoft.com/en-us/azure/application-insights/app-insights-proactive-diagnostics
50 https://docs.microsoft.com/en-us/azure/azure-monitor/app/alerts
51 https://docs.microsoft.com/en-us/azure/azure-monitor/app/app-map
52 https://docs.microsoft.com/en-us/azure/application-insights/app-insights-profiler
53 https://docs.microsoft.com/en-us/azure/application-insights/app-insights-usage-overview

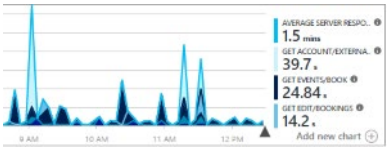
Diagnostic search for instance data[54]

Search and filter events such as requests, exceptions, dependency calls, log traces, and page views.



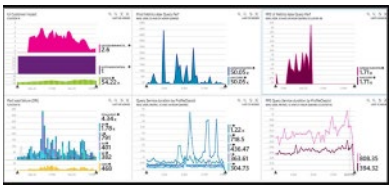
Metrics Explorer for aggregated data

Explore, filter, and segment aggregated data such as rates of requests, failures, and exceptions; response times, page load times.



Dashboards

Mash up data from multiple resources and share with others. Great for multi-component applications, and for continuous display in the team room.



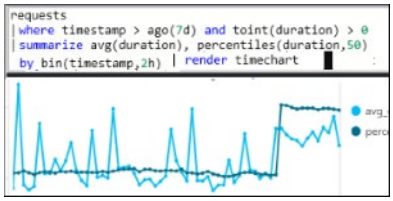
Live Metrics Stream

When you deploy a new build, watch these near-real-time performance indicators to make sure everything works as expected.

Analytics

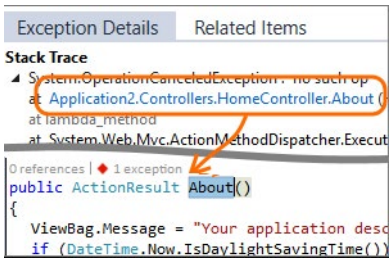
Answer tough questions about your app's performance and usage by using this powerful query language.

54 https://docs.microsoft.com/en-us/azure/application-insights/app-insights-diagnostic-search



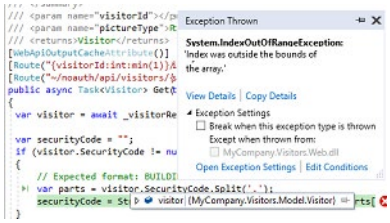
Visual Studio

See performance data in the code. Go to code from stack traces.



Snapshot debugger

Debug snapshots sampled from live operations, with parameter values.

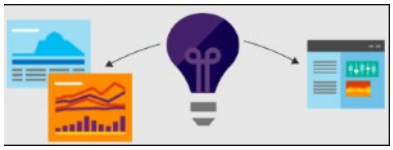


Power BI

Integrate usage metrics with other business intelligence.

REST API

Write code to run queries over your metrics and raw data.



Continuous export

Bulk export of raw data to storage as soon as it arrives.

How do I use Application Insights

Monitor

Install Application Insights in your app, set up **availability web tests**[55], and:

- Set up a **dashboard**[56] for your team room to keep an eye on load, responsiveness, and the performance of your dependencies, page loads, and AJAX calls.
- Discover which are the slowest and most failing requests.
- Watch **Live Stream**[57] when you deploy a new release, to know immediately about any degradation.

Detect, Diagnose

When you receive an alert or discover a problem:

- Assess how many users are affected.
- Correlate failures with exceptions, dependency calls and traces.
- Examine profiler, snapshots, stack dumps, and trace logs.

Build, Measure, Learn

Measure the effectiveness[58] of each new feature that you deploy.

- Plan to measure how customers use new UX or business features.
- Write custom telemetry into your code.
- Base the next development cycle on hard evidence from your telemetry.

Get started

Application Insights is one of the many services hosted within Microsoft Azure, and telemetry is sent there for analysis and presentation. So before you do anything else, you'll need a subscription to **Microsoft Azure**[59]. It's free to sign up, and if you choose the basic **pricing plan**[60] of Application Insights, there's no charge until your application has grown to have substantial usage. If your organization already has a subscription, they could add your Microsoft account to it.

55 https://docs.microsoft.com/en-us/azure/application-insights/app-insights-monitor-web-app-availability
56 https://docs.microsoft.com/en-us/azure/azure-monitor/app/app-insights-dashboards
57 https://docs.microsoft.com/en-us/azure/application-insights/app-insights-live-stream
58 https://docs.microsoft.com/en-us/azure/application-insights/app-insights-usage-overview
59 https://azure.com/
60 https://azure.microsoft.com/pricing/details/application-insights/

There are several ways to get started. Begin with whichever works best for you. You can add the others later.

- At run time: instrument your web app on the server. Avoids any update to the code. You need admin access to your server.
  - **IIS on-premises or on a VM**[61]
  - **Azure web app or VM**[62]
  - **J2EE**[63]
- At development time: add Application Insights to your code. Allows you to write custom telemetry and to instrument back-end and desktop apps.
  - **Visual Studio**[64] 2013 update 2 or later.
  - **Java**[65]
  - Node.js
  - **Other platforms**[66]
- **Instrument your web pages**[67] for page view, AJAX and other client-side telemetry.
- **Analyze mobile app usage**[68] by integrating with Visual Studio App Center.
- **Availability tests**[69] - ping your website regularly from our servers.

Application Insights

Application performance management (APM) is a discipline that includes all the tools and activities involved in observing how software and hardware are performing. These tools present that performance information in a form product owners and software development teams can use to make decisions. Application Insights is Microsoft Azure native APM Tool that is cross platform and specialises in providing a rich & intelligent performance management toolset for Azure hosted web apps.

In this tutorial we'll learn how to get started with App Insights. We'll cover,

- Adding App Insights to your dotnet core app
- Accessing App Insights from within the Azure Portal

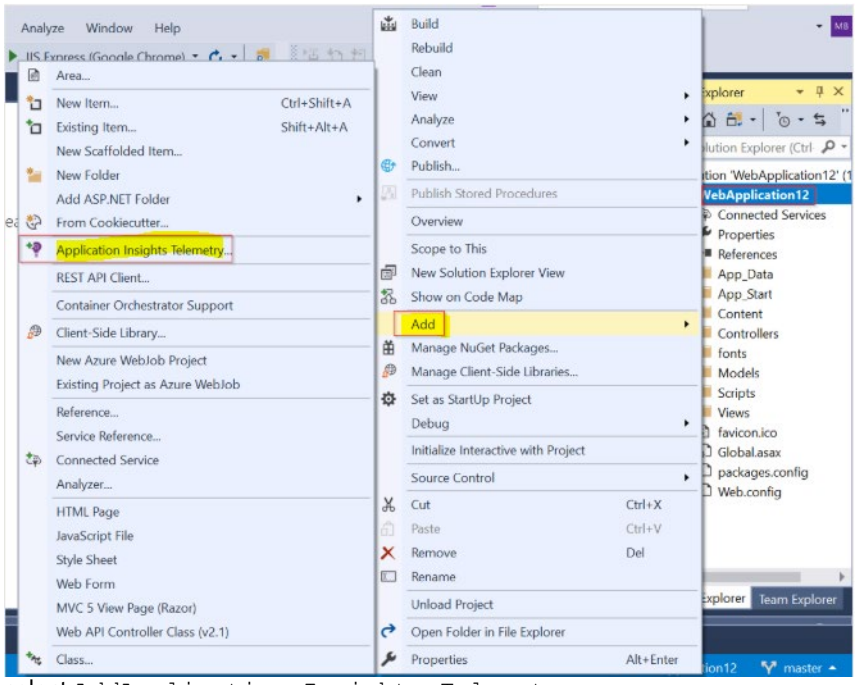
Getting Started

1. To add Application Insights to your ASP.NET website, you need to:

- Install Visual Studio 2019 for Windows with the following workloads:
- ASP.NET and web development (Do not uncheck the optional components)

61 https://docs.microsoft.com/en-us/azure/application-insights/app-insights-monitor-performance-live-website-now
62 https://docs.microsoft.com/en-us/azure/application-insights/app-insights-monitor-performance-live-website-now
63 https://docs.microsoft.com/en-us/azure/application-insights/app-insights-java-live
64 https://docs.microsoft.com/en-us/azure/azure-monitor/app/asp-net
65 https://docs.microsoft.com/en-us/azure/application-insights/app-insights-java-get-started
66 https://docs.microsoft.com/en-us/azure/application-insights/app-insights-platforms
67 https://docs.microsoft.com/en-us/azure/application-insights/app-insights-javascript
68 https://docs.microsoft.com/en-us/azure/application-insights/app-insights-mobile-center-quickstart
69 https://docs.microsoft.com/en-us/azure/application-insights/app-insights-monitor-web-app-availability

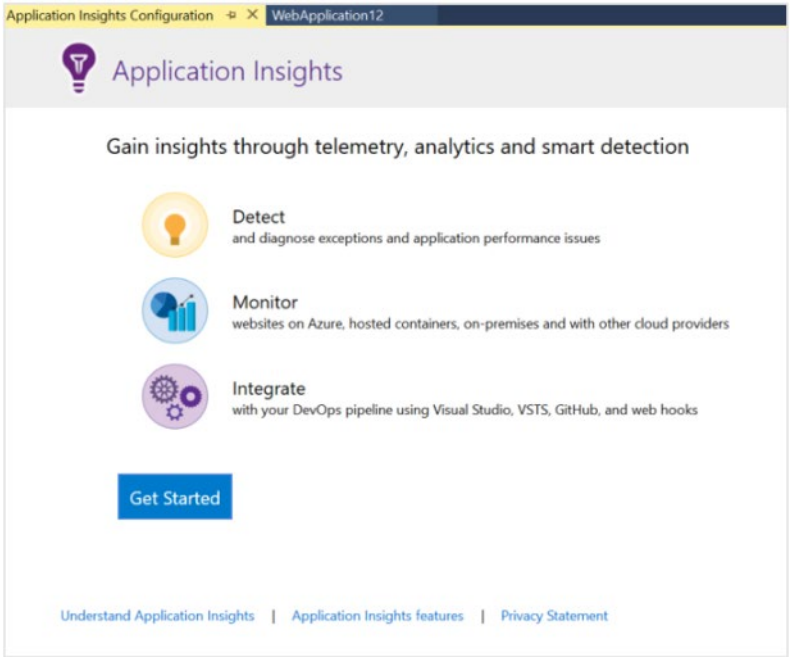
1. In Visual Studio create a new dotnet core project. Right click the project and from the context menu



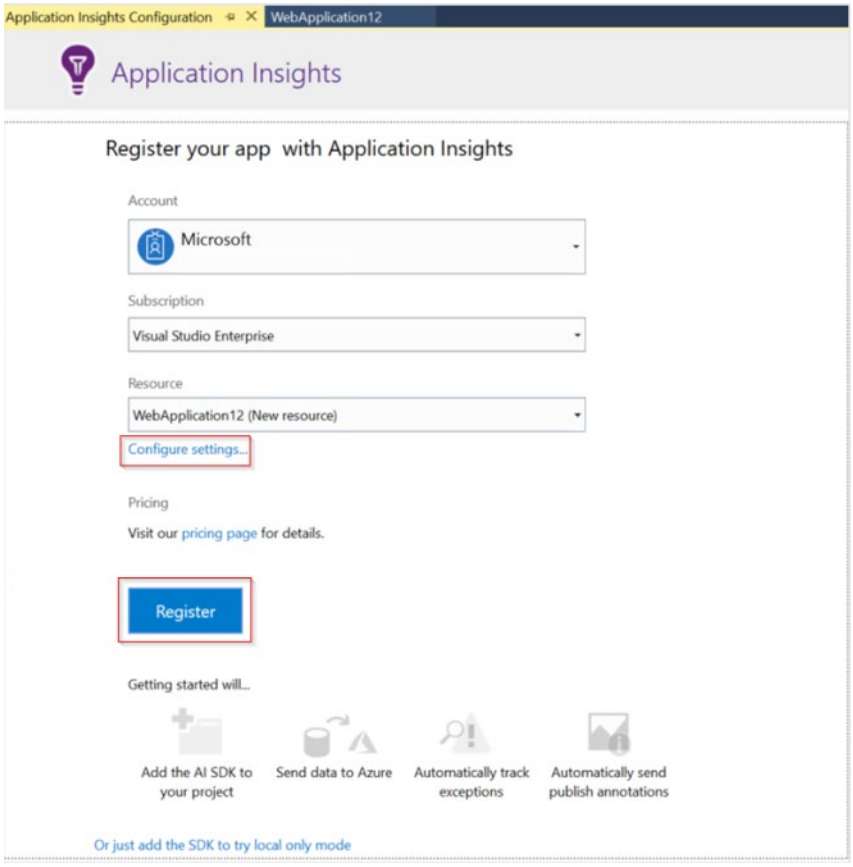
select `AddApplication Insights Telemetry`

(Depending on your Application Insights SDK version you may be prompted to upgrade to the latest SDK release. If prompted, select Update SDK.)

1. From the Application Insights configuration screen, click Get started to start setting up App Insights



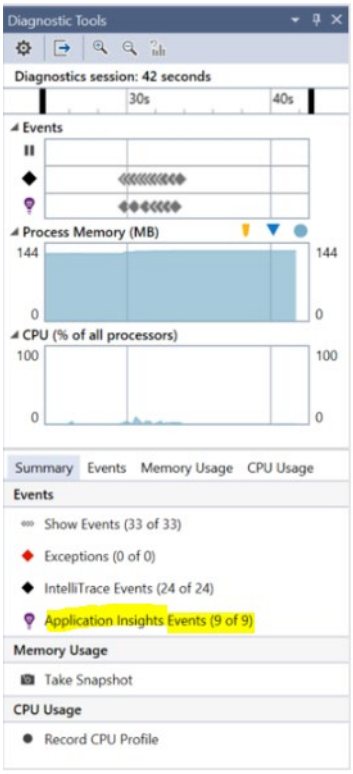
2. Choose to set up a new resource group and select the location where you want the telemetry data to be persisted.



Summary: So far we've added App Insights in a dotnet core application. The Application Insights getting started experience gives you the ability to create a new resource group in the desired location where the App Insights instance gets created. The instrumentation key for the app insights instance is injected into the application configuration automatically.

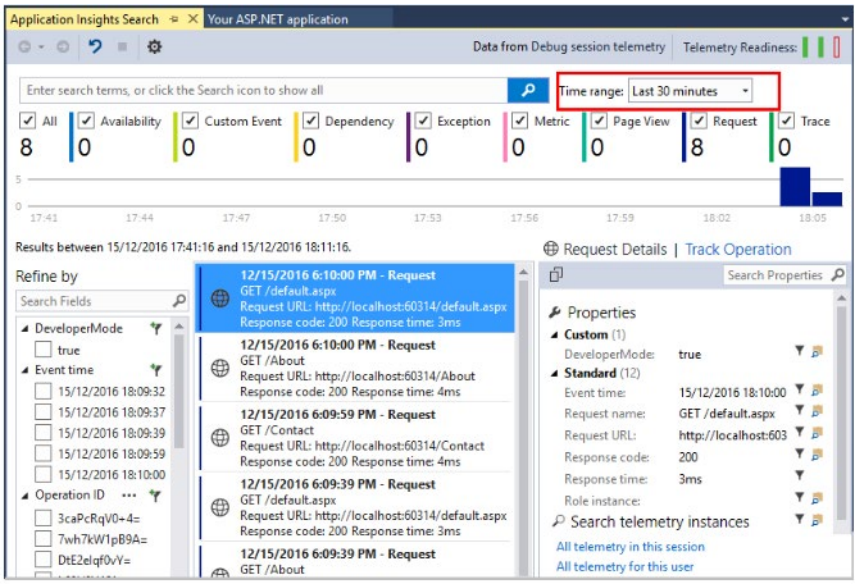
How to do it

1. Run your app with F5. Open different pages to generate some telemetry. In Visual Studio, you will see a count of the events that have been logged.



1. You can see your telemetry either in Visual Studio or in the Application Insights web portal. Search telemetry in Visual Studio to help you debug your app. Monitor performance and usage in the web portal when your system is live. In Visual Studio, to view Application Insights data. Select Solution Explorer > Connected Services > right-click Application Insights, and then click Search Live Telemetry.

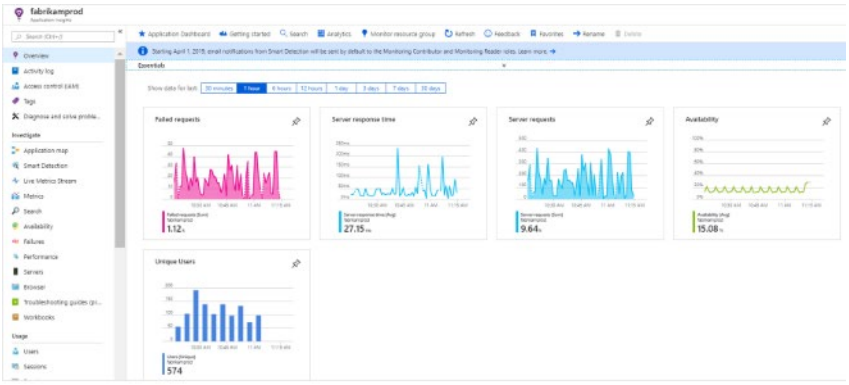
In the Visual Studio Application Insights Search window, you will see the data from your application for telemetry generated in the server side of your app. Experiment with the filters, and click any event to see more detail.



1. You can also see telemetry in the Application Insights web portal (unless you chose to install only the SDK). The portal has more charts, analytic tools, and cross-component views than Visual Studio. The portal also provides alerts.

Open your Application Insights resource. Either sign into the Azure portal and find it there, or select Solution Explorer > Connected Services > right-click Application Insights > Open Application Insights Portal and let it take you there.

The portal opens on a view of the telemetry from your app.



How it works

Application Insights configures an unique key (called AppInsights Key) in your application. This key is used by the Application Insights SDK to identify the Azure App Insights workspace the telemetry data needs to be uploaded into. The SDK and the key are merely used to pump the telemetry data points out of your application. The heavy lifting of data correlation, analysis and insights is done within Azure.

There's more

In this tutorial we learned how to get started by adding Application Insights into your dotnet core application. App Insights offers a wide range of features, you can learn more about these on **Microsoft docs**[70]

70 https://docs.microsoft.com/en-us/azure/azure-monitor/learn/dotnetcore-quick-start

Implement routing for mobile application crash report data

Introduction

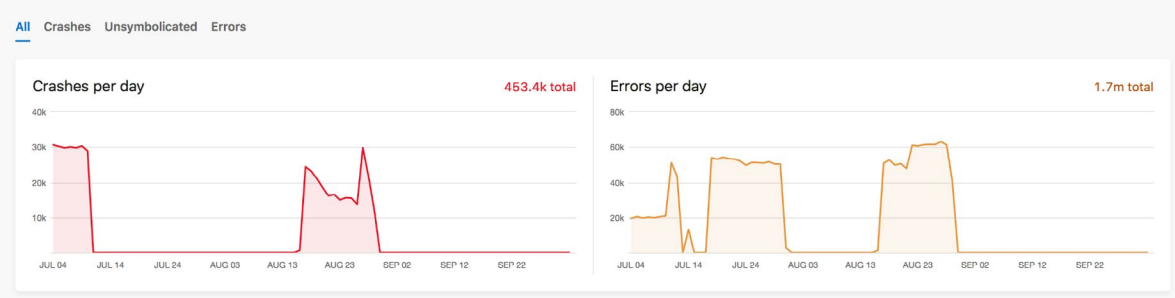
App Center Diagnostics is a cloud service that helps developers monitor the health of an application, delivering the data needed to understand what happens when an app fails during testing or in the wild. The App Center Diagnostics SDK collects information about crashes and errors in your apps and uploads them to the App Center portal for analysis by the development team - eliminating the guesswork about what really happened in the app when it failed.

Crashes
Crashes are what happens when a runtime exception occurs from an unexpected event that terminates the app. These are errors not handled by a try/catch block. When a crash occurs, App Center Crashes records the state of the app and device and automatically generates a crash log. These logs contain valuable information to help you fix the crash.

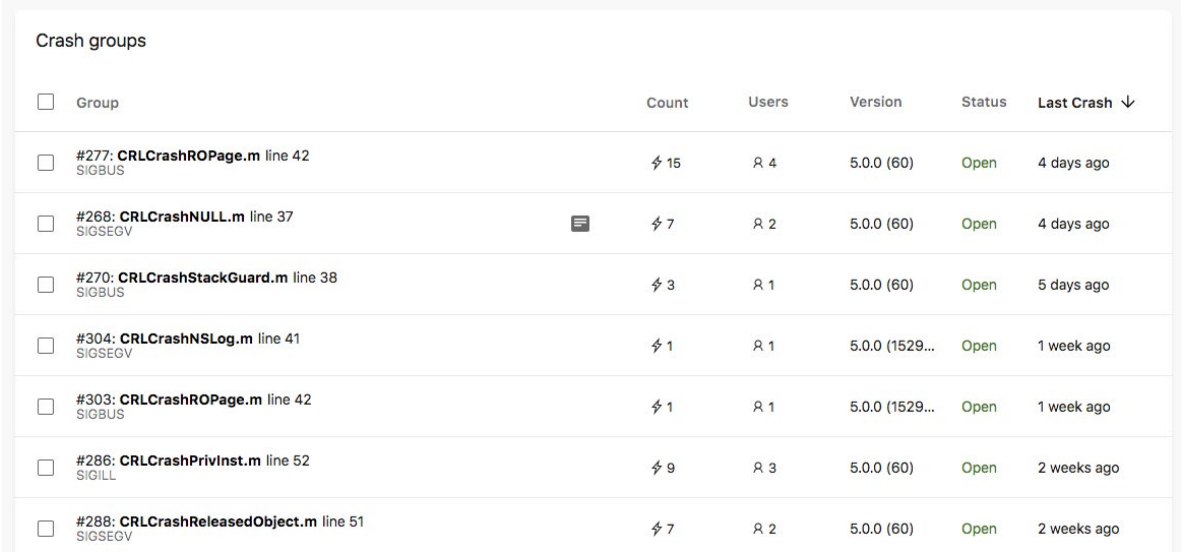
Crash and Errors Analytics
In App Center Diagnostics, you can view analytics data generated automatically by App Center to understand when a crash or error occurs in your app.

By default, App Center displays an app's crashes and errors per day in a side by side view.

Using the top-left tabs, drill down into Crashes and Errors. When you do this, the left chart indicates the number of crashes/errors per day, and the right chart shows the number of affected users. Filter the charts by app version, time frame and status for a more focused view.



Grouping
App Center Diagnostics groups crashes and errors by similarities, such as reason for the issue and where the issue occurred in the app. For each crash and error group, App Center displays the line of code that failed, the class or method name, file name, line number, crash or error type and message for you to better understand these groups at a glance. Select a group to view more information, and access a list of detailed issues reports and logs. This allows you to dive even deeper and use our feature set to better understand your app's behavior during a crash or an error.



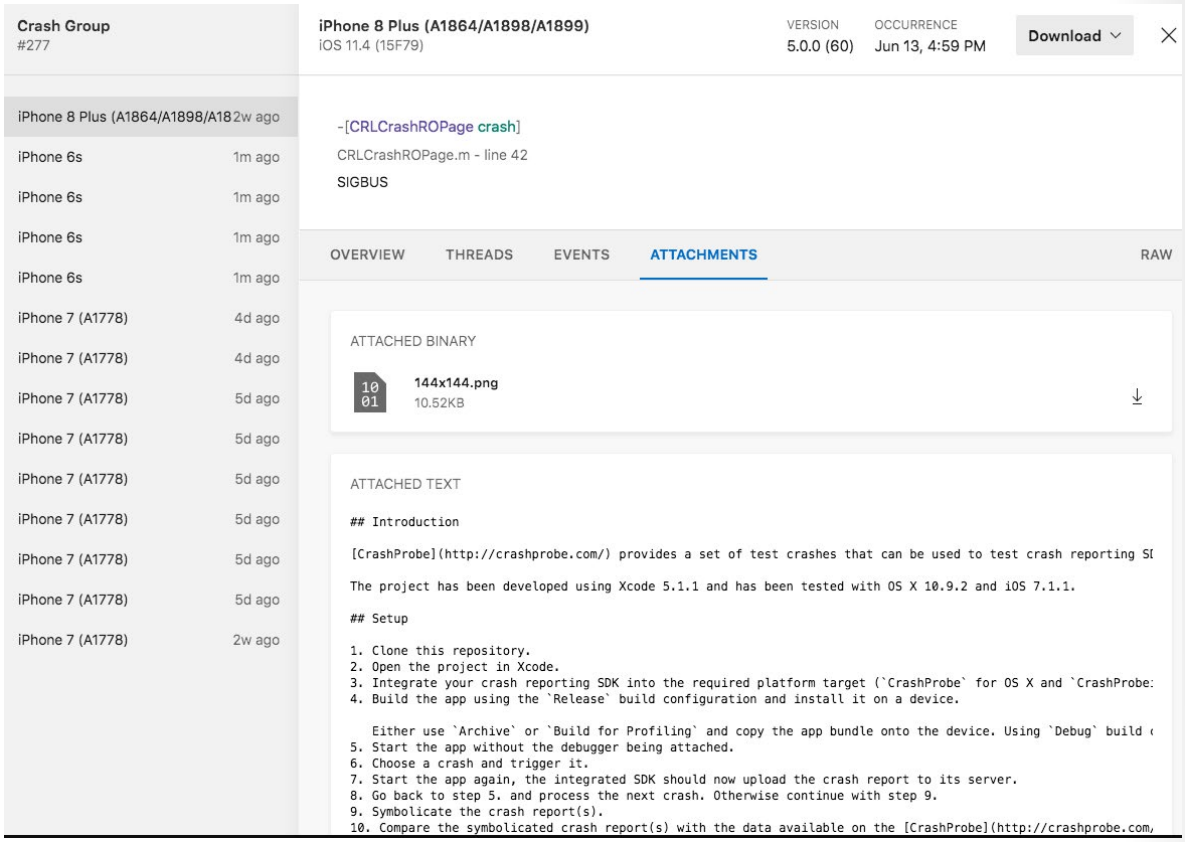
Attachments

In the App Center Diagnostics UI, you can attach, view and download one binary and one text attachment to your crash reports.

You can learn how to add attachments to your crash reports by reading the SDK Crashes documentation for your **Android**[71], **iOS**[72], **macOS**[73], **React Native**[74], **Xamarin**[75], and **Apache Cordova**[76] apps.

To view and download the attachments, select a crash group, a specific device report and then click on the attachments tab.

71 https://docs.microsoft.com/en-us/appcenter/sdk/crashes/android#add-attachments-to-a-crash-report
72 https://docs.microsoft.com/en-us/appcenter/sdk/crashes/ios#add-attachments-to-a-crash-report
73 https://docs.microsoft.com/en-us/appcenter/sdk/crashes/macos#add-attachments-to-a-crash-report
74 https://docs.microsoft.com/en-us/appcenter/sdk/crashes/react-native#add-attachments-to-a-crash-report
75 https://docs.microsoft.com/en-us/appcenter/sdk/crashes/xamarin#add-attachments-to-a-crash-report
76 https://docs.microsoft.com/en-us/appcenter/sdk/crashes/cordova#add-attachments-to-a-crash-report



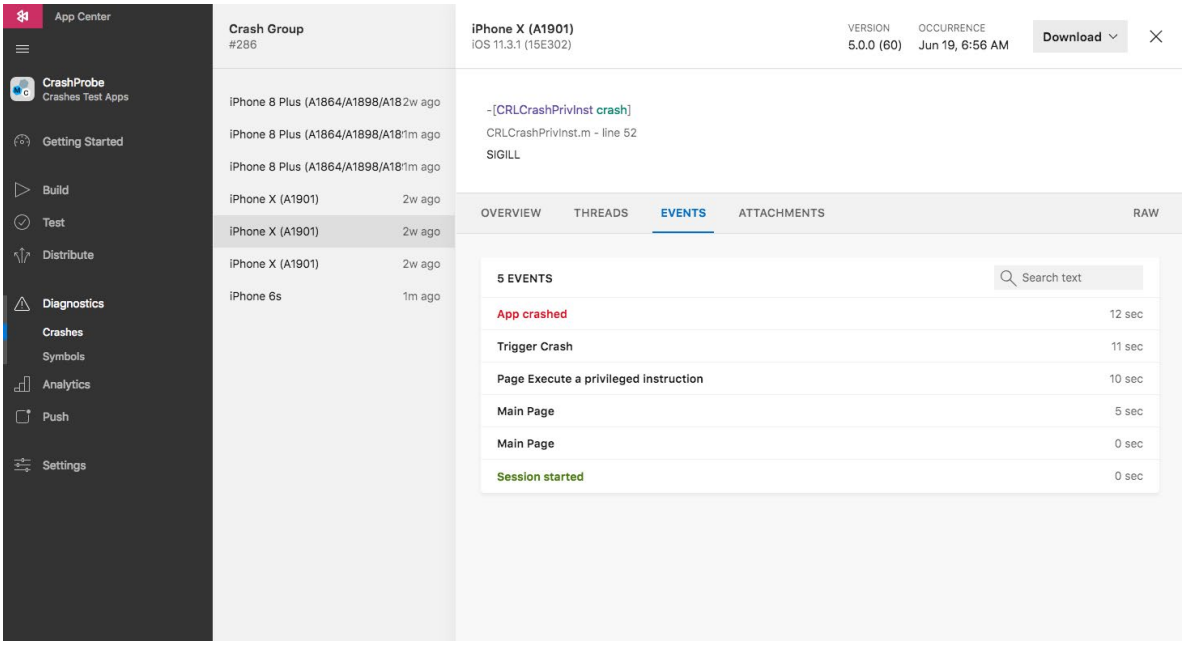
Events Before a Crash

Track events leading up to a crash to capture useful information about the state of your app.

To define a custom event, check out our **SDK Documentation**[77] for **Android**[78], **iOS**[79], **React Native**[80], **Xamarin**[81], **UWP**[82] and **macOS**[83].

To view events before a crash, select a crash group, a specific device report, and then click on the events tab.

77 https://docs.microsoft.com/en-us/appcenter/sdk/index
78 https://docs.microsoft.com/en-us/appcenter/sdk/analytics/android
79 https://docs.microsoft.com/en-us/appcenter/sdk/analytics/ios
80 https://docs.microsoft.com/en-us/appcenter/sdk/analytics/react-native
81 https://docs.microsoft.com/en-us/appcenter/sdk/analytics/xamarin
82 https://docs.microsoft.com/en-us/appcenter/sdk/getting-started/uwp
83 https://docs.microsoft.com/en-us/appcenter/sdk/analytics/macos



Configure Alerts

Stay on top of your crashes by configuring your App Center app definition settings to send an email when a new crash group is created. To configure these alerts:

1. Log into App Center and select your app
2. In the left menu, navigate to Settings
3. Click on Email Notifications
4. Select the box next to Crashes

Create a Bug Tracker

You can integrate third party bug tracker tools with App Center to stay informed and better manage your crashes. Read the **bug tracker documentation**[84] to learn how to get started.

App Center has bug tracker integration for the crashes service. Users can be quickly informed about critical App Center events within the tools that you use regularly in your day to day flow for a seamless experience. App Center supports bug trackers like Jira, Azure DevOps (formerly Visual Studio Team Services (VSTS)), and GitHub. Users need to have manager or developer permissions to be able to create and configure the bug tracker.

For Azure DevOps (formerly VSTS):

1. Login with your Azure DevOps credentials and click Accept when prompted on app authorization.
2. Select which Azure DevOps projects to integrate the bug tracker with and click Next.

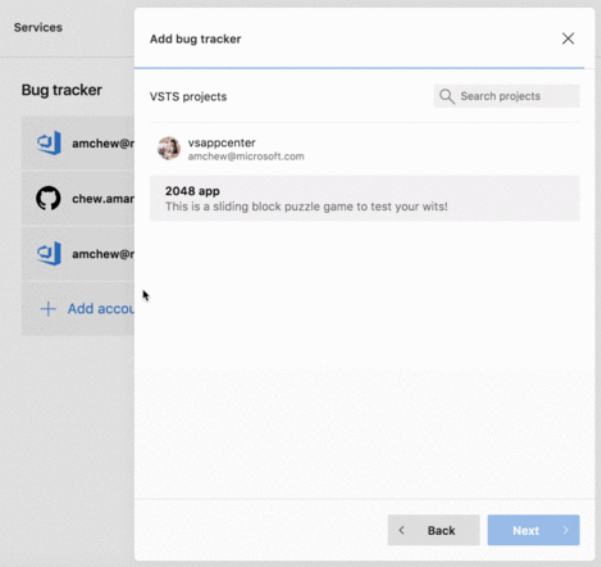
84 https://docs.microsoft.com/en-us/appcenter/dashboard/bugtracker/index

3. Under Add bug tracker, fill in the fields for Number of crashes, Area and Default Payload, and click Add:

   - Number of crashes is a threshold you can set for the minimum number of crashes to happen in a crash group before a ticket is created in Azure DevOps.
   - Default payload is an optional field to fill in for use in work items. For example,

{“System.IterationPath”: "Area\Iteration 1", “System.AssignedTo”: "Fabrikam"}.

Please refer to the **work item types API**[85] for additional information.



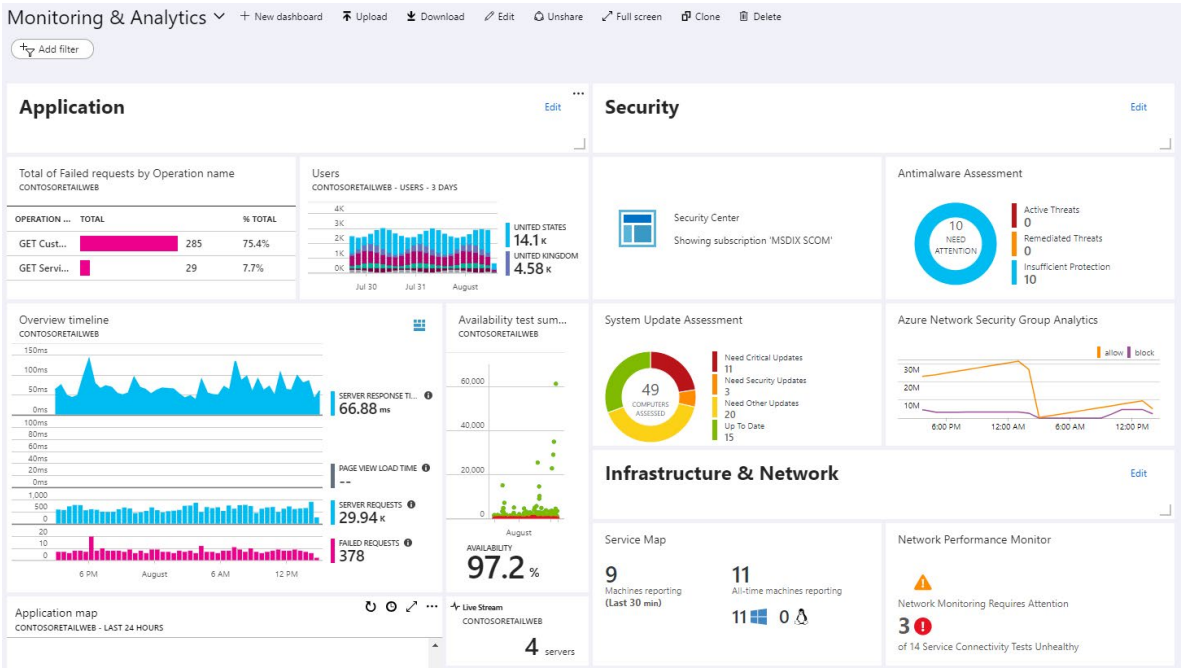
85 https://docs.microsoft.com/en-us/rest/api/vsts/wit/work%20item%20types

Develop monitoring and status dashboards

Azure Dashboards

Visualizations such as charts and graphs can help you analyze your monitoring data to drill-down on issues and identify patterns. Depending on the tool you use, you may also have the option to share visualizations with other users inside and outside of your organization.

Azure dashboards[86] are the primary dashboarding technology for Azure. They're particularly useful in providing single pane of glass over your Azure infrastructure and services allowing you to quickly identify important issues.



Advantages

- Deep integration into Azure. Visualizations can be pinned to dashboards from multiple Azure pages including metrics analytics, log analytics, and Application Insights.
- Supports both metrics and logs.
- Combine data from multiple sources including output from **Metrics explorer**[87], **Log Analytics queries**[88], and **maps**[89] and **availability**[90] in Application Insights.

86 https://docs.microsoft.com/en-us/azure/azure-portal/azure-portal-dashboards
87 https://docs.microsoft.com/en-us/azure/azure-monitor/platform/metrics-charts
88 https://docs.microsoft.com/en-us/azure/azure-monitor/log-query/log-query-overview
89 https://docs.microsoft.com/en-us/azure/azure-monitor/app/app-map
90 https://docs.microsoft.com/en-us/azure/azure-monitor/visualizations?toc=/azure/azure-monitor/toc.json

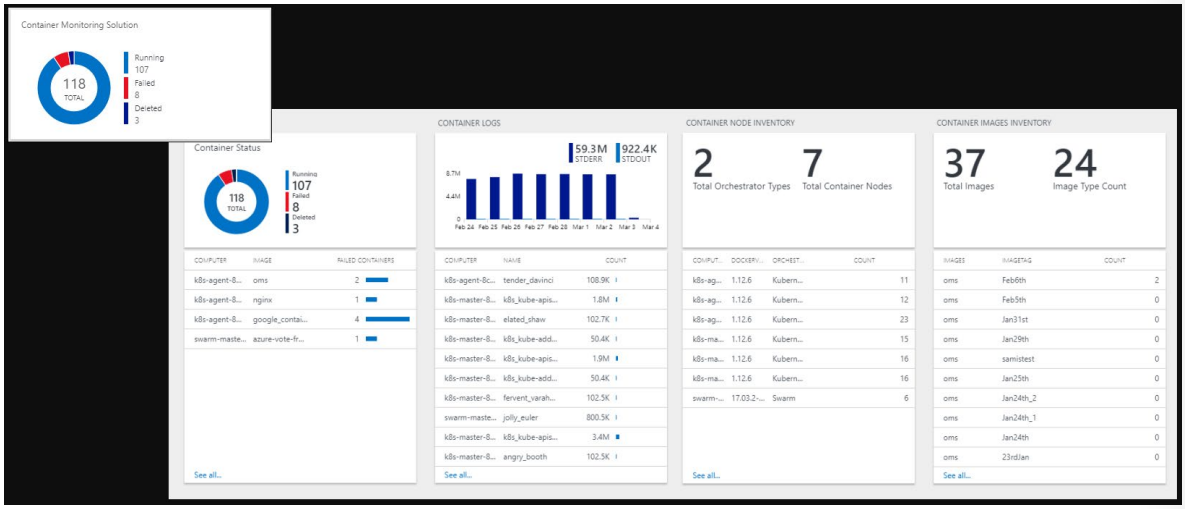
- Option for personal or shared dashboards. Integrated with Azure **role based authentication (RBAC)**[91].
- Automatic refresh. Metrics refresh depends on time range with minimum of five minutes. Logs refresh at one minute.
- Parametrized metrics dashboards with timestamp and custom parameters.
- Flexible layout options.
- Full screen mode.

Limitations

- Limited control over log visualizations with no support for data tables. Total number of data series is limited to 10 with further data series grouped under an *other* bucket.
- No custom parameters support for log charts.
- Log charts are limited to last 30 days.
- Log charts can only be pinned to shared dashboards.
- No interactivity with dashboard data.
- Limited contextual drill-down.

Azure Monitor Views

Views in Azure Monitor[92] allow you to create custom visualizations with log data. They are used by **monitoring solutions**[93] to present the data they collect.



91 https://docs.microsoft.com/en-us/azure/role-based-access-control/overview
92 https://docs.microsoft.com/en-us/azure/log-analytics/log-analytics-view-designer
93 https://docs.microsoft.com/en-us/azure/azure-monitor/insights/solutions

Advantages

- Rich visualizations for log data.
- Export and import views to transfer them to other resource groups and subscriptions.
- Integrates into Log Analytic management model with workspaces and monitoring solutions.
- **Filters**[94] for custom parameters.
- Interactive, supports multi-level drill-in (view that drills into another view)

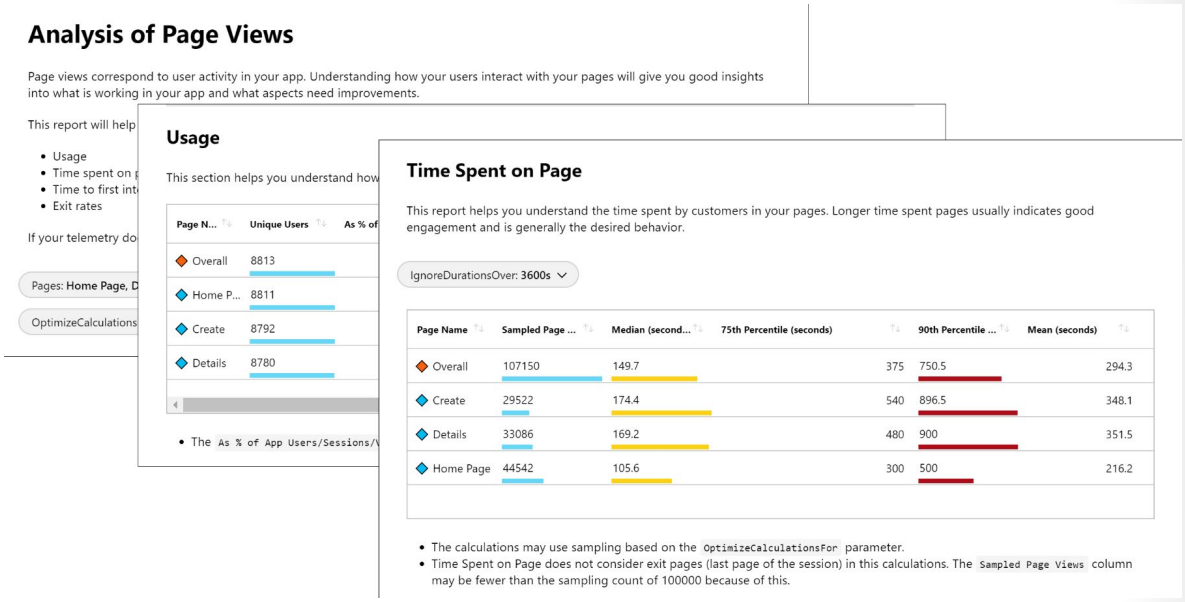
Limitations

- Supports logs but not metrics.
- No personal views. Available to all users with access to the workspace.
- No automatic refresh.
- Limited layout options.
- No support for querying across multiple workspaces or Application Insights applications.
- Queries are limited in response size to 8MB and query execution time of 110 seconds.

Application Insights Workbooks

Workbooks[95] are interactive documents that provide deep insights into your data, investigation, and collaboration inside the team. Specific examples where workbooks are useful are troubleshooting guides and incident postmortem.

94 https://docs.microsoft.com/en-us/azure/azure-monitor/platform/view-designer-filters
95 https://docs.microsoft.com/en-us/azure/application-insights/app-insights-usage-workbooks



Advantages

- Supports both metrics and logs.
- Supports parameters enabling interactive reports where selecting an element in a table will dynamically update associated charts and visualizations.
- Document-like flow.
- Option for personal or shared workbooks.
- Easy, collaborative-friendly authoring experience.
- Templates support public GitHub-based template gallery.

Limitations

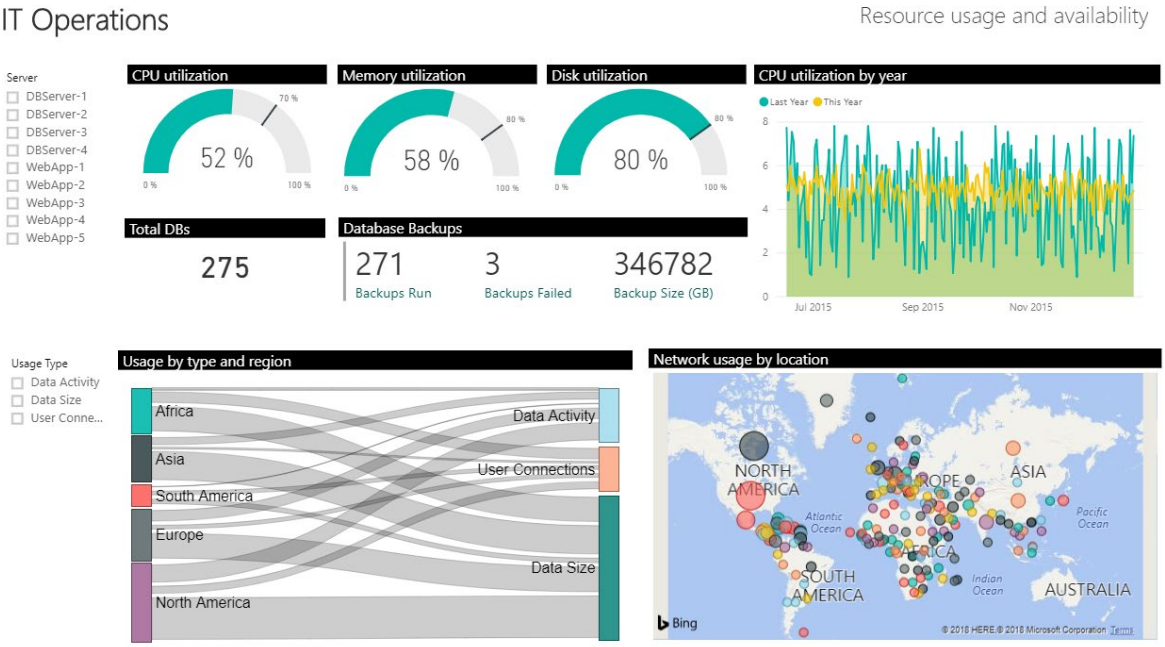
- No automatic refresh.
- No dense layout like dashboards, which make workbooks less useful as a single pane of glass. Intended more for providing deeper insights.

Power BI

Power BI[96] is particularly useful for creating business-centric dashboards and reports, as well as reports analyzing long-term KPI trends. You can **import the results of a log query**[97] into a Power BI dataset so

96 https://powerbi.microsoft.com/documentation/powerbi-service-get-started/
97 https://docs.microsoft.com/en-us/azure/log-analytics/log-analytics-powerbi

you can take advantage of its features such as combining data from different sources and sharing reports on the web and mobile devices.



Advantages

- Rich visualizations.
- Extensive interactivity including zoom-in and cross-filtering.
- Easy to share throughout your organization.
- Integration with other data from multiple data sources.
- Better performance with results cached in a cube.

Limitations

- Supports logs but not metrics.
- No Azure integration. Can't manage dashboards and models through Azure Resource Manager.
- Query results need to be imported into Power BI model to configure. Limitation on result size and refresh.
- Limited data refresh of eight times per day.

Grafana

Grafana[98] is an open platform that excels in operational dashboards. It's particularly useful for detecting and isolating and triaging operational incidents. You can add **Grafana Azure Monitor data source plugin**[99] to your Azure subscription to have it visualize your Azure metrics data.

Advantages

- Rich visualizations.
- Rich ecosystem of datasources.
- Data interactivity including zoom in.
- Supports parameters.

Limitations

- Supports metrics but not logs.
- No Azure integration. Can't manage dashboards and models through Azure Resource Manager.
- Cost to support additional Grafana infrastructure or additional cost for Grafana Cloud.

Build Your Own Custom Application

You can access data in log and metric data in Azure Monitor through their API using any REST client, which allows you to build your own custom websites and applications.

98 https://grafana.com/
99 https://docs.microsoft.com/en-us/azure/azure-monitor/platform/grafana-plugin

Advantages

- Complete flexibility in UI, visualization, interactivity, and features.
- Combine metrics and log data with other data sources.

Disadvantages

- Significant engineering effort required.

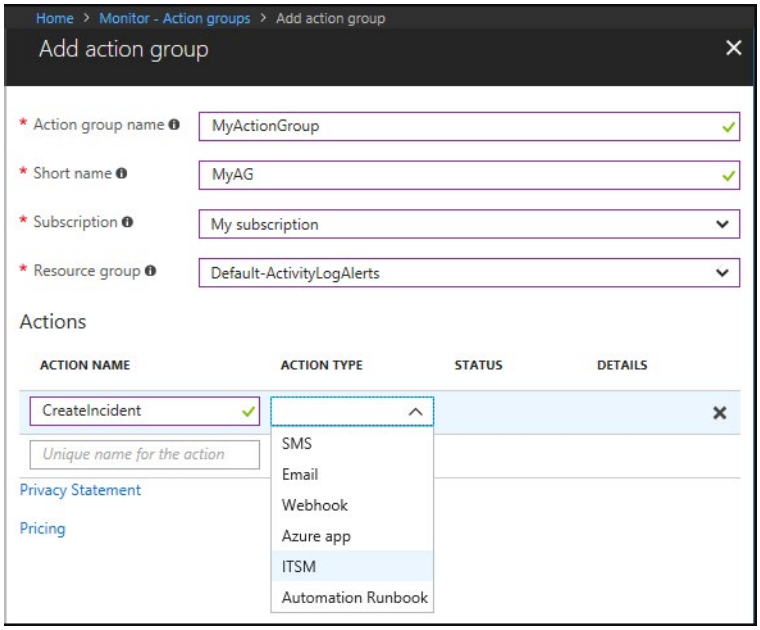
Integrate and configure ticketing systems

ITSM Connector

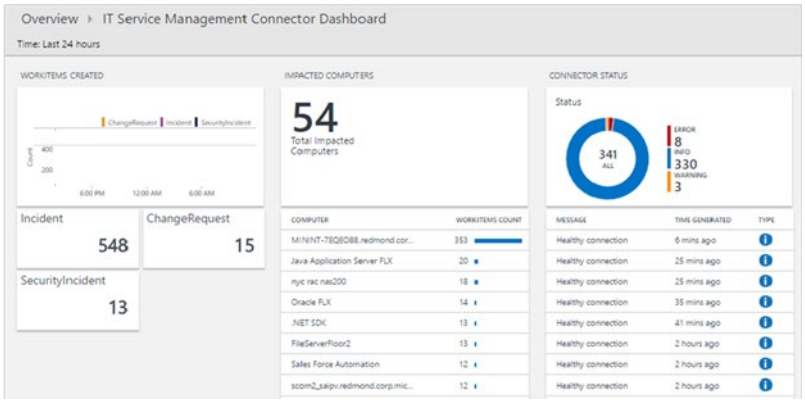
Customers use Azure monitoring tools to identify, analyze and troubleshoot issues. However, the work items related to an issue is typically stored in an ITSM tool. Instead of having to go back and forth between your ITSM tool and Azure monitoring tools, customers can now get all the information they need in one place. ITSMC will improve the troubleshooting experience and reduce the time it takes to resolve issues. IT Service Management Connector (ITSMC) for Azure provides bi-directional integration between Azure monitoring tools and your ITSM tools – ServiceNow, Provance, Cherwell, and System Center Service Manager. Specifically, you can use ITSMC to:

- Create or update work-items (Event, Alert, Incident) in the ITSM tools based on Azure alerts (Activity Log Alerts, Near Real-Time metric alerts and Log Analytics alerts)
- Pull the Incident and Change Request data from ITSM tools into Azure Log Analytics.

You can setup ITSMC by following the steps in our documentation. Once set up, you can send Azure alerts to ITSM tool using the ITSM action in Action groups.



You can also view your incident and change request data in Log Analytics to perform trend analysis or correlate it against operational data.



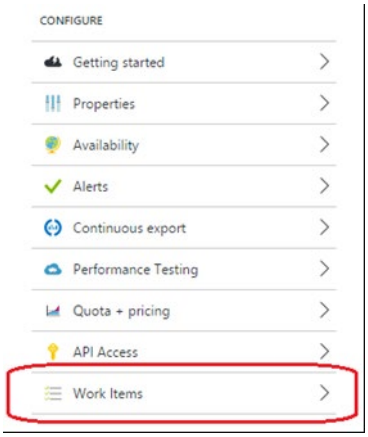
https://docs.microsoft.com/en-gb/azure/azure-monitor/platform/itsmc-overview

Integration with Azure Boards

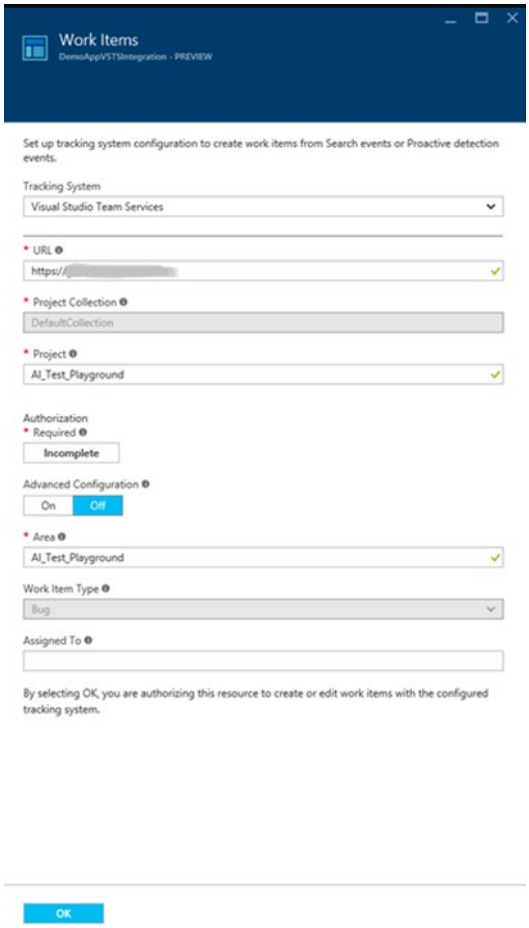
Work item integration functionality allows you to easily create work items in VSTS that have relevant Application Insights data embedded in them. It is very straightforward to configure this association and begin creating work items (this process should only take a minute or two).

Configuring work item integration

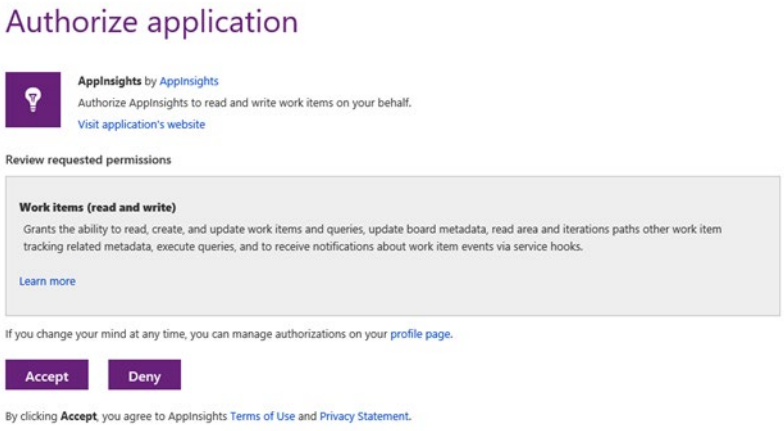
To configure work item integration for an Application Insights resource, simply navigate to your settings blade for that resource. You'll note that there is now a new item in the "Configure" section of the settings blade that says "Work Items."



Click on this, and the configuration blade for work items will open. All you need to do is fill out the information about the VSTS system to which you want to connect, along with the project where you want to write your work items:



Once that information is in place, you can click on the Authorization button, where you will be redirected to authorize access in your selected VSTS system so that work items can be written there:



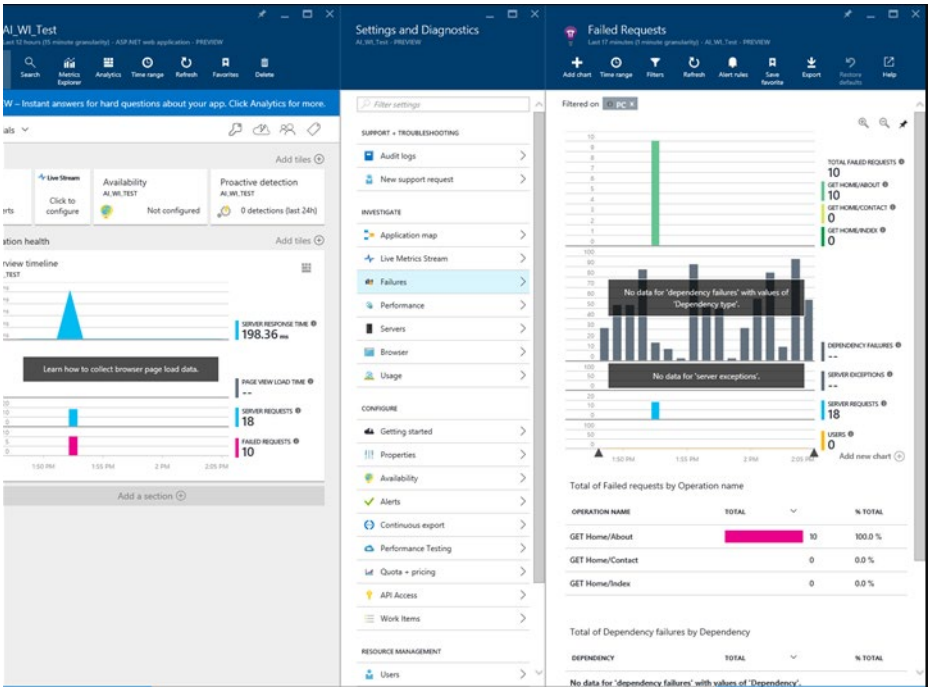
Once you've completed the authorization process, you can set defaults for "area path" and "assigned to." Only area path is required (if you haven't set up specific area paths in your project, that's ok. Just use the

name of the project, as it is the top-level area path. Click OK, and assuming you've entered everything correctly, you'll see a message stating "Validation Successful" and the blade will close. You're now ready to start creating work items!

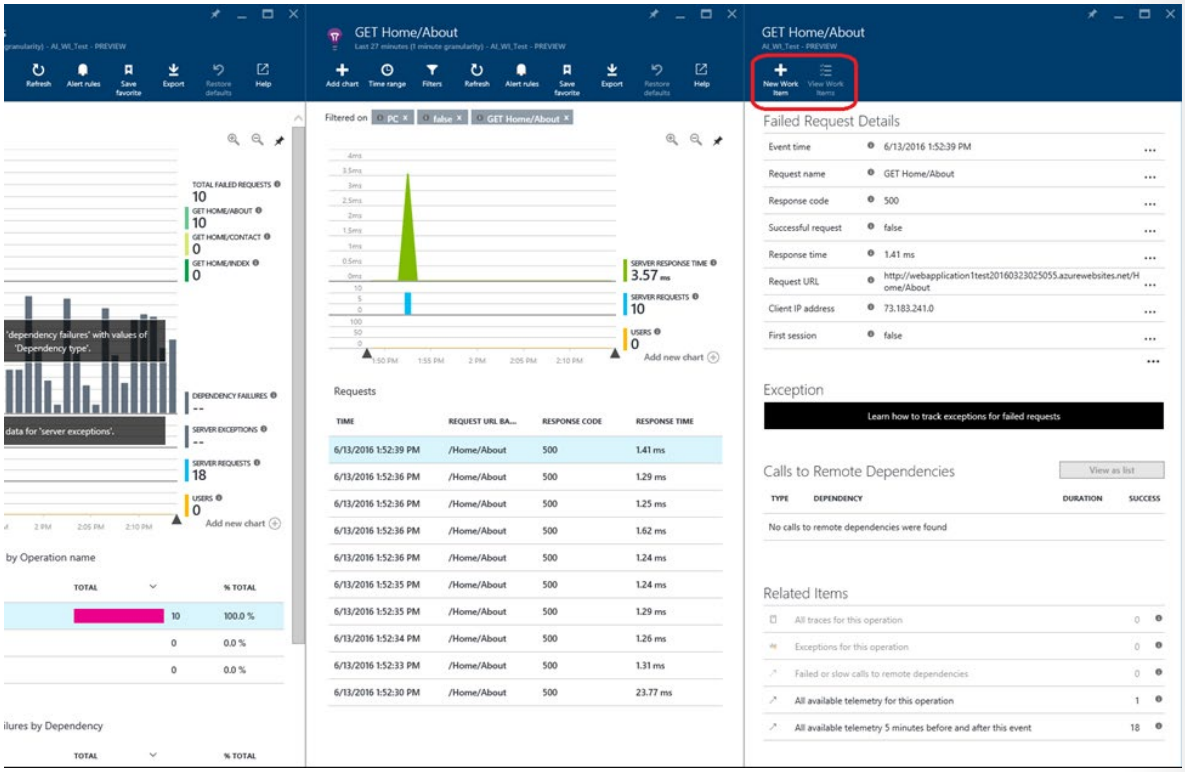
Creating work items

Creating work items from Application Insights is very easy. There are currently two locations from where you can create work items: Proactive detection and individual instances of activity (i.e., exceptions, failures, requests, etc.). I will show you a simple example of the latter, but the functionality is identical in either case.

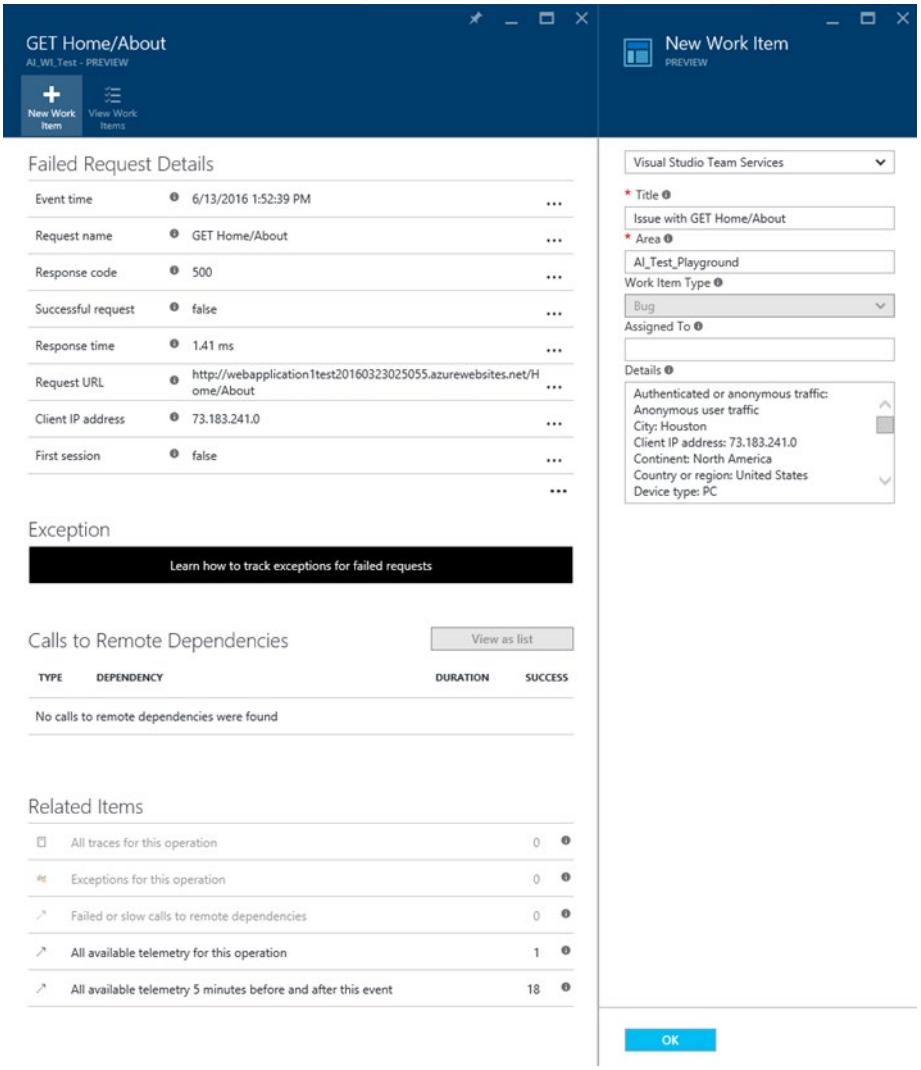
In this example, I'm looking at a test web app that I've published to Azure. I've started to drill into the activity for this app by looking at the Failures blade (but we could also get to this same information through the Search button or the Metrics Explorer):



We can see that I have a number of exceptions that fired when the user clicked on the Home/About tab on this web app. If I drill into this group of exceptions, I can see the list, and then choose an individual exception:



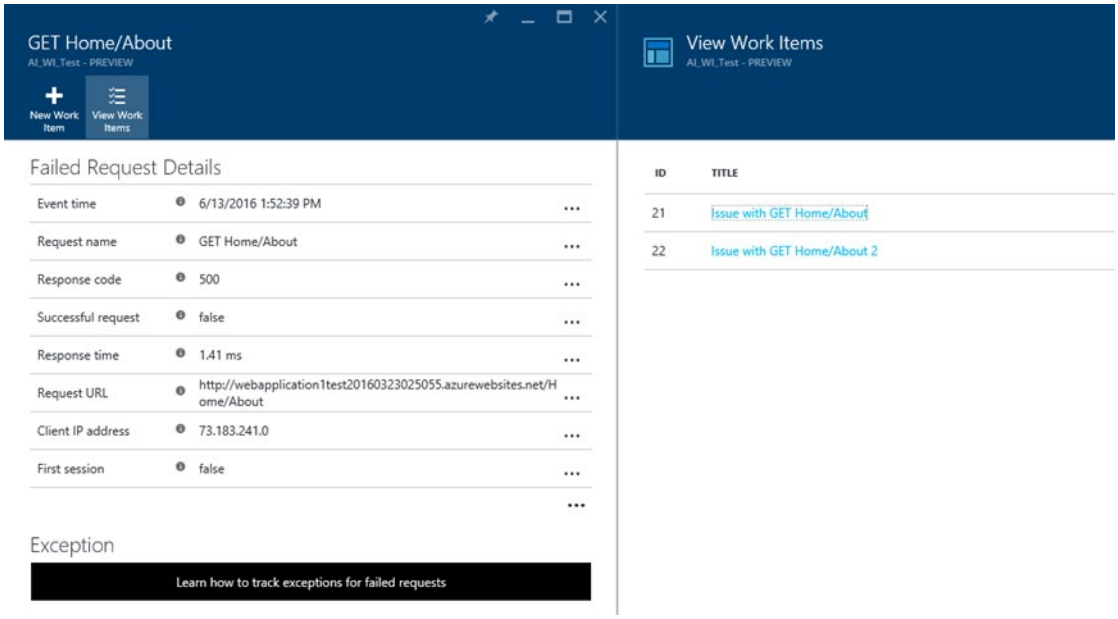
Looking at the detail blade for this exception, we see that there are now two buttons available at the top of the blade that read "New Work Item" and "View Work Items." To create a work item, I simply click on the first of these buttons, and it opens the new work item blade:



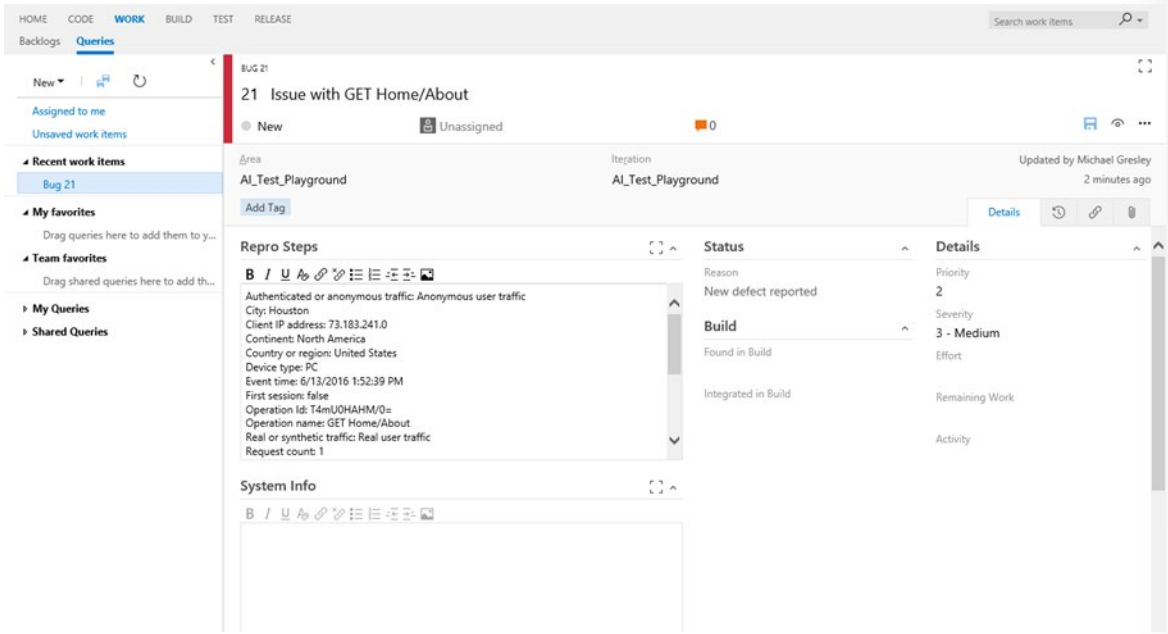
As you can see, just about everything you need in your average scenario has been filled out for you. The default values for "area path" and "assigned to" that you designated in the initial configuration are set, and all of the detail information we have available for this exception has been added to the details field. You can override the title, area path and assigned to fields in this blade if you wish, or you can add to the captured details. When you're ready to create your work item, just clicked on the "OK" button, and your work item will be written to VSTS.

Viewing work items

Once you've created one or more work items in Application Insights, you can easily view them in VSTS. If you are in the Azure portal, the detail blade for the event associated to the work item(s) will have the "View Work Items" button enabled. To see the list, simply click the button:



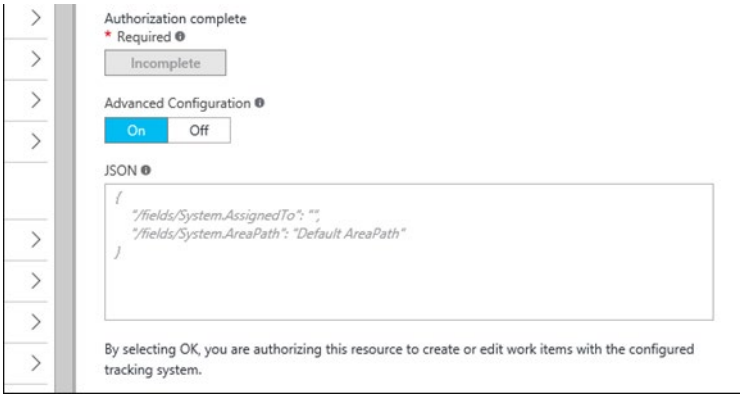
If you click the link for the work item that you want to view, it will open in VSTS:



Advanced Configuration

Some of you may have noticed that there is a switch on the configuration blade that is labeled "Advanced Configuration." This is additional functionality that we've provided to help you configure your ability to write to VSTS in scenarios where you've changed or extended some of the out-of-the-box settings. A good example of this is designating additional required fields. Currently, there is no way to handle this additional required mapping in the standard config, but you can handle it in advanced mode.

If you click on the switch, the controls at the bottom of the blade will change to look like this:



You can see that you are now given a JSON-based editing box where you can specify all of the particular settings/mappings that you might need to handle modifications to your VSTS project. Some sample text is filled in for you. In the near future, we plan to enhance this control with intellisense as well as publish some basic guidance to better understand the advanced configuration mode.

Next steps

We think that this is a good start to integrating work item functionality with Application Insights. But, please keep in mind that this is essentially the 1.0 version of this feature set. We have a lot of work planned, and you will see a significant evolution in this space over the upcoming months. Just for starters, let me outline a few of the things that we already have planned or are investigating:

- *Support for all work item types* – You probably noticed that the current feature set locks the work item type to just "bug." Logging bugs was our primary ask for this space, so that's where we started, but we certainly don't think that's where things should end. One of the more near-term changes that you will see is to handle all work item types for all supported processes in VSTS.
- *Links back to Application Insights* – It's great to be able to create a work item with App Insights data in it, but what happens when you're in your ALM system and looking at that item and want to quickly navigate back to the source of the work item in App Insights? We plan to add links to the work items in the very near future to make this as fast and easy as possible.
- *More flexible configuration* – Currently, our standard configuration only handles scenarios where the user has not significantly modified/extended their project in VSTS. Today, if you've made these kind of changes, you'll need to switch to advanced configuration mode. Going forward, we want to handle common things that people might change (e.g., making additional fields required, adding new fields) in the standard configuration wherever possible. This requires some updates from our friends on the VSTS team, but they are already working on some of these for us. Once they're available, we will begin to make the standard configuration more flexible. In the meantime (and in the future), you can always use the advanced configuration to overcome any limitations.
- *Multiple profiles* – Setting up a single configuration means that in shops where there are several ways in which users commonly create work items, the people creating work items from Application Insights would have to frequently override values. We plan to give users the capability to set up 1:n profiles,

with standard values specified for each so that when you want to create a work item with that particular profile, you can simply choose it from a drop-down list.

- *More sources of creation for work items* – We will continue to investigate (and take feedback on) other places in Application Insights where it makes sense to create work items.
- *Automatic creation of work items* – There are certainly scenarios we can imagine (and I'm sure you can too) where we might want a work item to be created for us based upon criteria. This is definitely on the radar, but we are spending some design time with this to limit possibilities of super-noisy or runaway work item creation. We believe that this is a powerful and convenient feature, but we want to reduce the potential for spamming the ALM system as much as possible.
- *Support for other ALM systems* – Hey, we think that VSTS is an awesome product, but we recognize that many of our users may use some other product for their ALM, and we want to meet people where they are. So, we are working on additional first-tier integrations of popular ALM products. We also plan to provide a pure custom configuration choice (similar to advanced config for VSTS) so that end users will be able to hook up Application Insights to virtually any ALM system.

Module 3 Implement and Manage Build Infrastructure

Site Reliability Engineering

What is Site Reliability Engineering

Software developers spend a lot of time chasing bugs and putting out production fires. Thanks to agile development, we are constantly shipping new code. By-products of constant change are constant issues with performance, software defects, and other issues that eat up our time. Web applications that receive even a modest amount of traffic require constant care and feeding. This includes overseeing deployments, monitoring overall performance, reviewing error logs, and troubleshooting software defects. These tasks have traditionally been handled by a mixture of lead developers, development management, system administrators and more often than not, nobody. The problem is that these critical tasks lacked a clear owner … until now.

What is site reliability engineering?

Site reliability engineering (SRE) empowers software developers to own the ongoing daily operation of their applications in production. The goal is to bridge the gap between the development team that wants to ship things as fast as possible and the operations team that doesn't want anything to blow up in production. In many organizations, you could argue that site reliability engineering eliminates much of the IT operations workload related to application monitoring. It shifts the responsibility to be part of the development team itself.

"Fundamentally, it's what happens when you ask a software engineer to design an operations function." – Niall Murphy, Google

Site reliability engineers typically spend up to 50% of their time dealing with the daily care and feeding of software applications. They spend the rest of their time writing code like any other software developer would.

A key skill of a software reliability engineer is that they have a deep understanding of the application, the code, and how it runs, is configured, and scales. That knowledge is what makes them so valuable at also monitoring and supporting it as a site reliability engineer.

Some of the typical responsibilities of a site reliability engineer:

- Proactively monitor and review application performance
- Handle on-call and emergency support
- Ensure software has good logging and diagnostics
- Create and maintain operational runbooks
- Help triage escalated support tickets
- Work on feature requests, defects and other development tasks
- Contribute to overall product roadmap

Perform live site reviews and capture feedback for system outages

The concept of site reliability engineering started in 2003 within Google. As Google continued to grow and scale to become the massive company they are today, they encountered many of their own growing pains. Their challenge was how to support large-scale systems while also introducing new features continuously.

To accomplish the goal, they created a new role that had the dual purpose of developing new features while also ensuring that production systems ran smoothly. Site reliability engineering has grown significantly within Google and most projects have site reliability engineers as part of the team. Google now has over 1,500 site reliability engineers.

Site reliability engineering vs DevOps

So, I know what you are thinking … how does site reliability engineering compare to DevOps?

DevOps is a more recent movement, designed to help organizations' IT department move in agile and performant ways. It builds a healthy working relationship between the Operations staff and Dev team, allowing each to see how their work influences and affects the other. By combining knowledge and effort, DevOps should produce a more robust, reliable, agile product.

Both SRE and DevOps are methodologies addressing organizations' needs for production operation management. But the differences between the two doctrines are quite significant: While DevOps raise problems and dispatch them to Dev to solve, the SRE approach is to find problems and solve some of them themselves. While DevOps teams would usually choose the more conservative approach, leaving the production environment untouched unless absolutely necessary, SREs are more confident in their ability to maintain a stable production environment and push for rapid changes and software updates. Not unlike the DevOps team, SREs also thrive on a stable production environment, but one of the SRE team's goals is to improve performance and operational efficiency.

For other companies like us who were born in the cloud and heavily use PaaS services, I believe they will also see site reliability engineering as the missing element to their development team success. For larger companies or companies who don't use the cloud, I could see them using both DevOps and site reliability engineering. DevOps practices can help ensure IT helps rack, stack, configure, and deploy the servers and applications. The site reliability engineers can then handle the daily operation of the applications. They

also work as a fast feedback loop to the entire team about how the application is performing and running in production.

Site reliability engineering skills

The type of skills needed will vary wildly based on your type of application, how and where it is deployed, and how it is monitored. For organizations using Serverless such as Azure PaaS in-depth knowledge of Windows or Linux systems management isn't much of a priority. However, it may be really critical to teams if you are using servers for deployments.

The other key skills for a good site reliability engineer are more focused on application monitoring and diagnostics. You want to hire people who are good problem solvers and have a knack for finding problems. Experience with application performance management tools like App Insights, New Relic, and others would be really valuable. They should be well versed at application logging best practices and exception handling.

The future of site reliability engineering

Software developers are increasingly taking a larger role in deployments, production operations, and application monitoring. The tools available today make it extremely easy to deploy our applications and monitor them. Things like PaaS and application monitoring solutions make it easy for developers to own their projects from ideation all the way to production.

While IT operations will always exist in most medium to large enterprises. Their type of work will continue to change because of the cloud, PaaS, containers, and other technologies.

Analyze telemetry to establish a baseline

When would I get a notifcation

Application Insights[1] automatically analyzes the performance of your web application, and can warn you about potential problems. You might be reading this because you received one of our smart detection notifications.

This feature requires no special setup, other than configuring your app for Application Insights (on **ASP.NET**[2], **Java**[3], or Node.js, and in **web page code**[4]). It is active when your app generates enough telemetry.

When would I get a smart detection notification?[5]

Application Insights has detected that the performance of your application has degraded in one of these ways:

- Response time degradation - Your app has started responding to requests more slowly than it used to. The change might have been rapid, for example because there was a regression in your latest deployment. Or it might have been gradual, maybe caused by a memory leak.
- Dependency duration degradation - Your app makes calls to a REST API, database, or other dependency. The dependency is responding more slowly than it used to.
- Slow performance pattern - Your app appears to have a performance issue that is affecting only some requests. For example, pages are loading more slowly on one type of browser than others; or requests are being served more slowly from one particular server. Currently, our algorithms look at page load times, request response times, and dependency response times.

Smart Detection requires at least 8 days of telemetry at a workable volume in order to establish a baseline of normal performance. So, after your application has been running for that period, any significant issue will result in a notification.

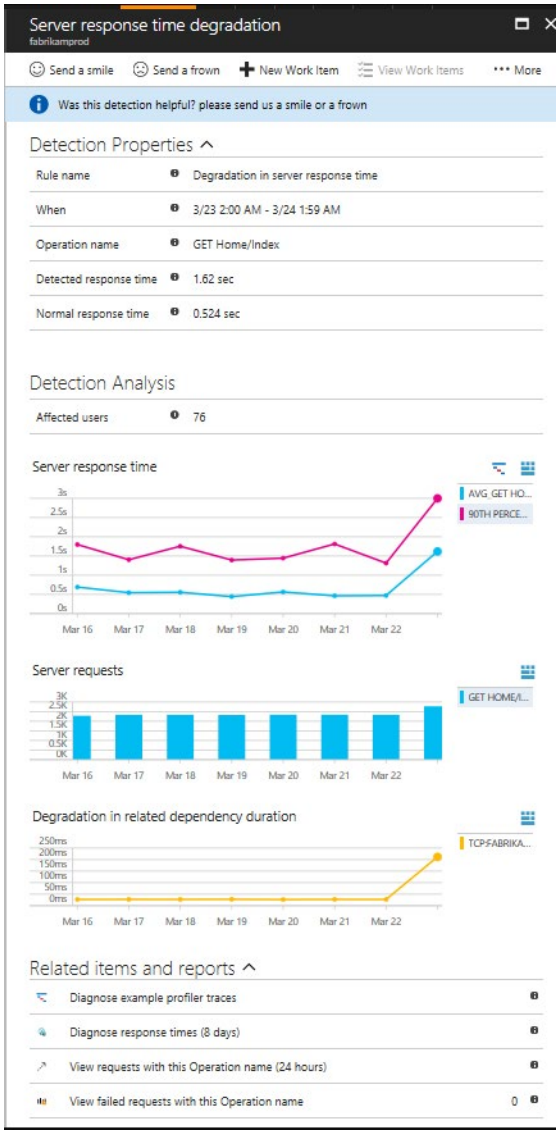
Does my app definitely have a problem

No, a notification doesn't mean that your app definitely has a problem. It's simply a suggestion about something you might want to look at more closely.

How do I fix it

The notifications include diagnostic information. Here's an example:

1 https://docs.microsoft.com/en-us/azure/application-insights/app-insights-overview
2 https://docs.microsoft.com/en-us/azure/azure-monitor/app/asp-net
3 https://docs.microsoft.com/en-us/azure/application-insights/app-insights-java-get-started
4 https://docs.microsoft.com/en-us/azure/application-insights/app-insights-javascript
5 https://docs.microsoft.com/en-us/azure/application-insights/app-insights-proactive-performance-diagnostics#when-would-i-get-a-smart-detection-notification

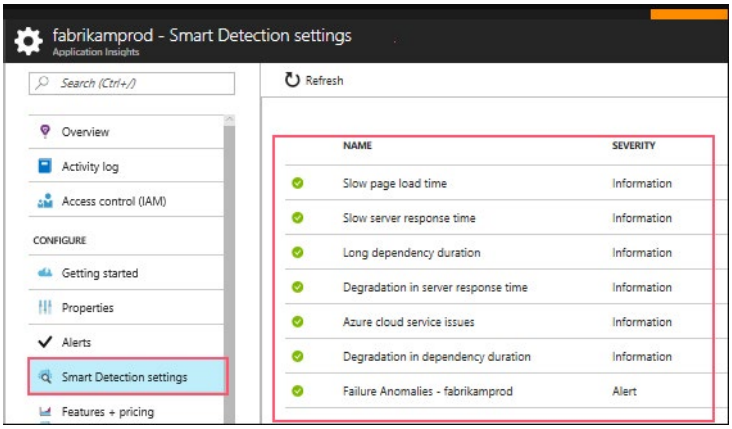


1. Triage. The notification shows you how many users or how many operations are affected. This can help you assign a priority to the problem.
2. Scope. Is the problem affecting all traffic, or just some pages? Is it restricted to particular browsers or locations? This information can be obtained from the notification.
3. Diagnose. Often, the diagnostic information in the notification will suggest the nature of the problem. For example, if response time slows down when request rate is high, that suggests your server or dependencies are overloaded. Otherwise, open the Performance blade in Application Insights. There, you will find **Profiler**[6] data. If exceptions are thrown, you can also try the **snapshot debugger**[7].

6 https://docs.microsoft.com/en-us/azure/application-insights/app-insights-profiler
7 https://docs.microsoft.com/en-us/azure/application-insights/app-insights-snapshot-debugger

Configure Email Notifications

1. Smart Detection notifications are enabled by default and sent to those who have **owners, contributors and readers access to the Application Insights resource**[8]. To change this, either click Configure in the email notification, or open Smart Detection settings in Application Insights.



- You can use the unsubscribe link in the Smart Detection email to stop receiving the email notifications.

Emails about Smart Detections performance anomalies are limited to one email per day per Application Insights resource. The email will be sent only if there is at least one new issue that was detected on that day. You won't get repeats of any message.

How can I improve performance?

- Slow and failed responses are one of the biggest frustrations for web site users, as you know from your own experience. So, it's important to address the issues.

Triage[9]

- First, does it matter? If a page is always slow to load, but only 1% of your site's users ever have to look at it, maybe you have more important things to think about. On the other hand, if only 1% of users open it, but it throws exceptions every time, that might be worth investigating. Use the impact statement (affected users or % of traffic) as a general guide, but be aware that it isn't the whole story.

8 https://docs.microsoft.com/en-us/azure/application-insights/app-insights-resources-roles-access-control
9 https://docs.microsoft.com/en-us/azure/application-insights/app-insights-proactive-performance-diagnostics#triage

Gather other evidence to confirm. Consider the parameters of the issue. If it's geography-dependent, set up **availability tests**[10] including that region: there might simply be network issues in that area.

Diagnose slow page loads[11]

- Where is the problem? Is the server slow to respond, is the page very long, or does the browser have to do a lot of work to display it? Open the Browsers metric blade. The segmented display of browser page load time shows where the time is going.

    - If Send Request Time is high, either the server is responding slowly, or the request is a post with a lot of data. Look at the **performance metrics**[12] to investigate response times.
    - Set up **dependency tracking**[13] to see whether the slowness is due to external services or your database.
    - If Receiving Response is predominant, your page and its dependent parts - JavaScript, CSS, images and so on (but not asynchronously loaded data) are long. Set up an **availability test**[14], and be sure to set the option to load dependent parts. When you get some results, open the detail of a result and expand it to see the load times of different files.
    - High Client Processing time suggests scripts are running slowly. If the reason isn't obvious, consider adding some timing code and send the times in trackMetric calls.

Improve slow pages[15]

- There's a web full of advice on improving your server responses and page load times, so we won't try to repeat it all here. Here are a few tips that you probably already know about, just to get you thinking:

    - Slow loading because of big files: Load the scripts and other parts asynchronously. Use script bundling. Break the main page into widgets that load their data separately. Don't send plain old HTML for long tables: use a script to request the data as JSON or other compact format, then fill the table in place. There are great frameworks to help with all this. (They also entail big scripts, of course.)
    - Slow server dependencies: Consider the geographical locations of your components. For example, if you're using Azure, make sure the web server and the database are in the same region. Do queries retrieve more information than they need? Would caching or batching help?
    - Capacity issues: Look at the server metrics of response times and request counts. If response times peak disproportionately with peaks in request counts, it's likely that your servers are stretched.

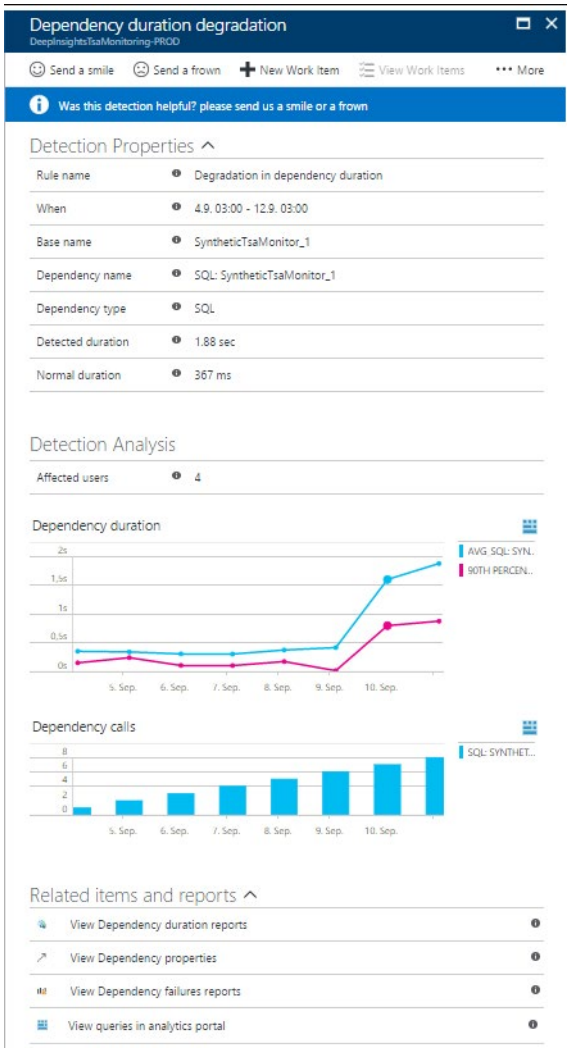
10 https://docs.microsoft.com/en-us/azure/application-insights/app-insights-monitor-web-app-availability
11 https://docs.microsoft.com/en-us/azure/application-insights/app-insights-proactive-performance-diagnostics#diagnose-slow-page-loads
12 https://docs.microsoft.com/en-us/azure/application-insights/app-insights-web-monitor-performance#metrics
13 https://docs.microsoft.com/en-us/azure/azure-monitor/app/asp-net-dependencies
14 https://docs.microsoft.com/en-us/azure/application-insights/app-insights-monitor-web-app-availability
15 https://docs.microsoft.com/en-us/azure/application-insights/app-insights-proactive-performance-diagnostics#improve-slow-pages

Server Response Time Degradation

- The response time degradation notification tells you:
  - The response time compared to normal response time for this operation.
  - How many users are affected.
  - Average response time and 90th percentile response time for this operation on the day of the detection and 7 days before.
  - Count of this operation requests on the day of the detection and 7 days before.
  - Correlation between degradation in this operation and degradations in related dependencies.
  - Links to help you diagnose the problem.
    - Profiler traces to help you view where operation time is spent (the link is available if Profiler trace examples were collected for this operation during the detection period).
    - Performance reports in Metric Explorer, where you can slice and dice time range/filters for this operation.
    - Search for this call to view specific call properties.
    - Failure reports - If count > 1 this mean that there were failures in this operation that might have contributed to performance degradation.

Dependency Duration Degradation

- Modern application more and more adopt micro services design approach, which in many cases leads to heavy reliability on external services. For example, if your application relies on some data platform or even if you build your own bot service you will probably relay on some cognitive services provider to enable your bots to interact in more human ways and some data store service for bot to pull the answers from. Example dependency degradation notification:



Notice that it tells you:

- The duration compared to normal response time for this operation
- How many users are affected
- Average duration and 90th percentile duration for this dependency on the day of the detection and 7 days before
- Number of dependency calls on the day of the detection and 7 days before
- Links to help you diagnose the problem

    - Performance reports in Metric Explorer for this dependency
    - Search for this dependency calls to view calls properties
    - Failure reports - If count > 1 this means that there were failed dependency calls during the detection period that might have contributed to duration degradation.

- Open Analytics with queries that calculate this dependency duration and count

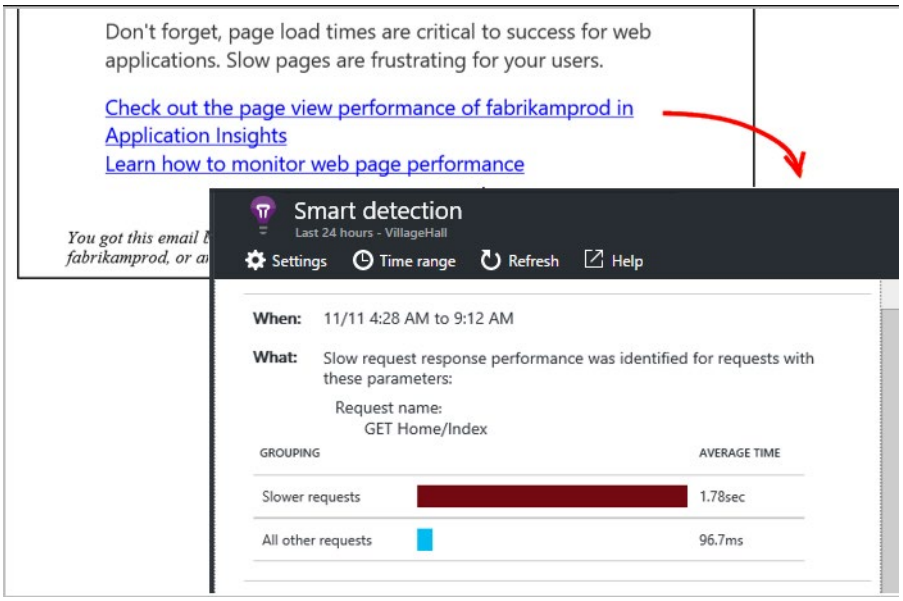
Smart Detection of slow performing patterns

Application Insights finds performance issues that might only affect some portion of your users, or only affect users in some cases. For example, notification about pages load is slower on one type of browser than on other types of browsers, or if requests are served more slowly from a particular server. It can also discover problems associated with combinations of properties, such as slow page loads in one geographical area for clients using particular operating system.

Anomalies like these are very hard to detect just by inspecting the data, but are more common than you might think. Often they only surface when your customers complain. By that time, it's too late: the affected users are already switching to your competitors!

Currently, our algorithms look at page load times, request response times at the server, and dependency response times.

You don't have to set any thresholds or configure rules. Machine learning and data mining algorithms are used to detect abnormal patterns.



- When shows the time the issue was detected.
- What describes:
  - The problem that was detected;
  - The characteristics of the set of events that we found displayed the problem behavior.
- The table compares the poorly-performing set with the average behavior of all other events.

Click the links to open Metric Explorer and Search on relevant reports, filtered on the time and properties of the slow performing set.

Modify the time range and filters to explore the telemetry.

For more information, view the video **Application Performance Management with Azure Application Insights**[16].

16 https://channel9.msdn.com/Events/Connect/2016/112/player

Perform ongoing tuning to reduce meaningless or non-actionable alerts

Monitoring and alerting

Monitoring and alerting enables a system to tell us when it's broken, or perhaps to tell us what's about to break. When the system isn't able to automatically fix itself, we want a human to investigate the alert, determine if there's a real problem at hand, mitigate the problem, and determine the root cause of the problem. Unless you're performing security auditing on very narrowly scoped components of a system, you should never trigger an alert simply because "something seems a bit weird." When you are reviewing existing alerts or writing new alerting rules, consider these things to keep your alerts relevant and your on-call rotation happier:

- Alerts that trigger call-out should be urgent, important, actionable, and real.
- They should represent either ongoing or imminent problems with your service.
- Err on the side of removing noisy alerts – over-monitoring is a harder problem to solve than under-monitoring.
- You should almost always be able to classify the problem into one of: availability & basic functionality; latency; correctness (completeness, freshness and durability of data); and feature-specific problems.
- Symptoms are a better way to capture more problems more comprehensively and robustly with less effort.
- Include cause-based information in symptom-based pages or on dashboards, but avoid alerting directly on causes.
- The further up your serving stack you go, the more distinct problems you catch in a single rule. But don't go so far you can't sufficiently distinguish what's going on.
- If you want a quiet oncall rotation, it's imperative to have a system for dealing with things that need timely response, but are not imminently critical.

Monitor for your users

I call this "symptom-based monitoring," in contrast to "cause-based monitoring". Do your users care if your MySQL servers are down? No, they care if their queries are failing. Do your users care if a support (i.e. non-serving-path) binary is in a restart-loop? No, they care if their features are failing. Do they care if your data push is failing? No, they care about whether their results are fresh.

Users, in general, care about a small number of things:

- Basic availability and correctness. no "Oops!", no 500s, no non-responding requests or half-loaded pages or missing Javascript or CSS or images or videos. - Anything that breaks the core service in some way should be considered unavailability.
- Latency. Fast. Fast. Fast. Also, fast.
- Completeness/freshness/durability. Your users data should be safe, should come back when you ask, and search indices should be up-to-date.
- Even if it is temporarily unavailable, users should have complete faith that it's coming back.
- Features. Your users care that all the features of the service work—you should be monitoring for anything that is an important aspect of your service even if it's not core functionality/availability.

That's pretty much it. There's a subtle but important difference between database servers being unavailable and user data being unavailable. The former is a proximate cause, the latter is a symptom. You can't always cleanly distinguish these things, particularly when you don't have a way to mimic the client's perspective (e.g. a blackbox probe or monitoring their perspective directly). But when you can, you should.

Cause-based alerts are bad (but sometimes necessary)

"But," you might say, "I know database servers that are unreachable results in user data unavailability." That's fine. Alert on the data unavailability. Alert on the symptom: the 500, the Oops!, the whitebox metric that indicates that not all servers were reached from the database's client. Why?

- You're going to have to catch the symptom anyway. Maybe it can happen because of network disconnection, or CPU contention, or myriad other problems you haven't thought of yet. So you have to catch the symptom.
- Once you catch the symptom and the cause, you have redundant alerts; these need separate tuning, and result in either duplication or complicated dependency trees
- The allegedly inevitable result is not always inevitable: maybe your database servers are unavailable because you're turning up a new instance or turning down an old one. Or maybe a feature was added to do fast-failover of requests, and so you don't care anymore about a single server's availability. Sure, you can catch all these cases with increasingly complicated rules, but why bother? The failure mode is more bogus pages, more confusion, and more tuning, with no gain, and less time spent on fixing the alerts that matter.

But sometimes they're necessary. There's (often) no symptoms to "almost" running out of quota or memory or disk I/O, etc., so you want rules to know you're walking towards a cliff. Use these sparingly; don't write cause-based rules that trigger on call alerts for symptoms you can catch otherwise.

Tickets, Reports and Email

One way or another, you have some alerts that need attention soon, but not right now. You can refer to these as "sub-critical alerts".

- Bug or ticket-tracking systems can be useful. Having alerts open a bug can work out great, as long as multiple firings of the same alert get correctly threaded into a single ticket/bug. This system fails if there's no accountability for triaging and closing bugs; if the alert-opened bugs might go unseen for weeks, this clearly fails as a way of dealing with sub-critical alerts before they become critical! It also fails if your team is simply overloaded or is not assigning enough people to deal with followup; you need to be honest about how much time this is consuming, or you'll fall further and further behind.
- A daily (or more frequent) report can work too. One way this can work is to write sub-critical rules that are long-lived (e.g. "the database is over 90% full" or "we've served over 1000 very slow requests in the last day"), and send out a report periodically that shows all currently-firing rules. Again, without a system of accountability this amounts to less-spammy email alerts, so make sure the oncall person (or someone else) is designated to triage these every day (or every shift hand-off, or whatever works).
- Every alert should be tracked through a workflow system. Don't only dump them into an email list or IRC channel. In general, this quickly turns into specialized "foo-alerts" mailing lists or channels so that they can be summarily ignored. Except as a brief (usually days, at most weeks) period to vet that a

new rule will not page too often, it's almost always a bad idea. It's also easy to ignore the volume of these alerts, and suddenly some old, mis-tuned rule is firing every minute for all of your thousand application servers, clogging up mailboxes. Oops.

The underlying point is to create a system that still has accountability for responsiveness, but doesn't have the high cost of waking someone up, interrupting their dinner, or preventing snuggling with a significant other.

Playbooks

Playbooks (or runbooks) are an important part of an alerting system; it's best to have an entry for each alert or family of alerts that catch a symptom, which can further explain what the alert means and how it might be addressed.

In general, if your playbook has a long detailed flow chart, you're potentially spending too much time documenting what could be wrong and too little time fixing it—unless the root causes are completely out of your control or fundamentally require human intervention (like calling a vendor). The best playbooks I've seen have a few notes about exactly what the alert means, and what's currently interesting about an alert ("We've had a spate of power outages from our widgets from VendorX; if you find this, please add it to Bug 12345 where we're tracking things for patterns".) Most such notes should be ephemeral, so a wiki or similar is a great tool.

Tracking & Accountability

Track your on-call alerts, and all your other alerts. If an on-call is firing and people just say "I looked, nothing was wrong", that's a pretty strong sign that you need to remove the rule, or demote it or collect data in some other way. Alerts that are less than 50% accurate are broken; even those that are false positives 10% of the time merit more consideration.

Having a system in place (e.g. a weekly review of all triggered on-call alerts, and quarterly statistics) can help keep a handle on the big picture of what's going on, and tease out patterns that are lost when the pager is handed from one human to the next.

When to seek the exception from the rule?

Here are some great reasons to break the above guidelines:

- You have a known cause that actually sits below the noise in your symptoms. For example, if your service has 99.99% availability, but you have a common event that causes 0.001% of requests to fail, you can't alert on it as a symptom (because it's in the noise) but you can catch the causing event. It might be worth trying to trickle this information up the stack, but maybe it really is simplest just to alert on the cause.
- You can't monitor at the spout, because you lose data resolution. For example, maybe you tolerate some handlers/endpoints/backends/URLs being pretty slow (like a credit card validation compared to browsing items for sale) or low-availability (like a background refresh of an inbox). At your load balancers, this distinction may be lost. Walk down the stack and alert from the highest place where you have the distinction.
- Your symptoms don't appear until it's too late, like you've run out of quota. Of course, you need to page before it's too late, and sometimes that means finding a cause to page on (e.g. usage > 80% and will run out in < 4h at the growth rate of the last 1h). But if you can do that, you should also be able to find a similar cause that's less urgent (e.g. quota > 90% and will run out in < 4d at the growth rate

of the last 1d) that will catch most cases, and deal with that as a ticket or email alert or daily problem report, rather than the last-ditch escalation that a page represents.

- Your alert setup sound more complex than the problems they're trying to detect. Sometimes they will be. The goal should be to tend towards simplicity, robust, self-protecting systems (how did you not notice that you were running out of quota? Why can't that data go somewhere else?) In the long term, they should trend towards simplicity, but at any given time the local optimum may be relatively complex rules to keep things quiet and accurate.

Analyze alerts to establish a baseline

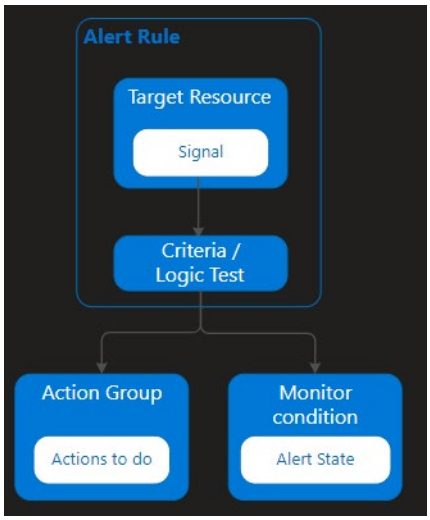
What are alerts in Microsoft Azure?

Alerts proactively notify you when important conditions are found in your monitoring data. They allow you to identify and address issues before the users of your system notice them.

This article discusses the unified alert experience in Azure Monitor, which now includes Log Analytics and Application Insights. The **previous alert experience**[17] and alert types are called classic alerts. You can view this older experience and older alert type by clicking on View classic alerts at the top of the alert page.

Overview[18]

The diagram below represents the flow of alerts.



Alert rules are separated from alerts and the action that are taken when an alert fires.

Alert rule - The alert rule captures the target and criteria for alerting. The alert rule can be in an enabled or a disabled state. Alerts only fire when enabled.

The key attributes of an alert rule are:

Target Resource - Defines the scope and signals available for alerting. A target can be any Azure resource. Example targets: a virtual machine, a storage account, a virtual machine scale set, a Log Analytics work-

17 https://docs.microsoft.com/en-us/azure/azure-monitor/platform/alerts-overview
18 https://docs.microsoft.com/en-us/azure/azure-monitor/platform/alerts-overview#overview

space, or an Application Insights resource. For certain resources (like Virtual Machines), you can specify multiple resources as the target of the alert rule.

Signal - Signals are emitted by the target resource and can be of several types. Metric, Activity log, Application Insights, and Log.

Criteria - Criteria is combination of Signal and Logic applied on a Target resource. Examples:

- Percentage CPU > 70%
- Server Response Time > 4 ms
- Result count of a log query > 100

Alert Name – A specific name for the alert rule configured by the user

Alert Description – A description for the alert rule configured by the user

Severity – The severity of the alert once the criteria specified in the alert rule is met. Severity can range from 0 to 4.

Action - A specific action taken when the alert is fired. For more information, see **Action Groups**[19].

What you can alert on

You can alert on metrics and logs as described in **monitoring data sources**[20]. These include but are not limited to:

- Metric values
- Log search queries
- Activity Log events
- Health of the underlying Azure platform
- Tests for web site availability

Manage alerts

You can set the state of an alert to specify where it is in the resolution process. When the criteria specified in the alert rule is met, an alert is created or fired, it has a status of *New*. You can change the status when you acknowledge an alert and when you close it. All state changes are stored in the history of the alert.

The following alert states are supported.

State	Description
New	The issue has just been detected and has not yet been reviewed.
Acknowledged	An administrator has reviewed the alert and started working on it.

19 https://docs.microsoft.com/en-us/azure/azure-monitor/platform/action-groups
20 https://docs.microsoft.com/en-us/azure/azure-monitor/platform/data-sources-reference

State	Description
Closed	The issue has been resolved. After an alert has been closed, you can reopen it by changing it to another state.

Alert state is different and independent of the monitor condition. Alert state is set by the user. Monitor condition is set by the system. When an alert fires, the alert's monitor condition is set to *fired*. When the underlying condition that caused the alert to fire clears, the monitor condition is set to *resolved*. The alert state isn't changed until the user changes it. Learn **how to change the state of your alerts and smart groups**[21].

Smart groups

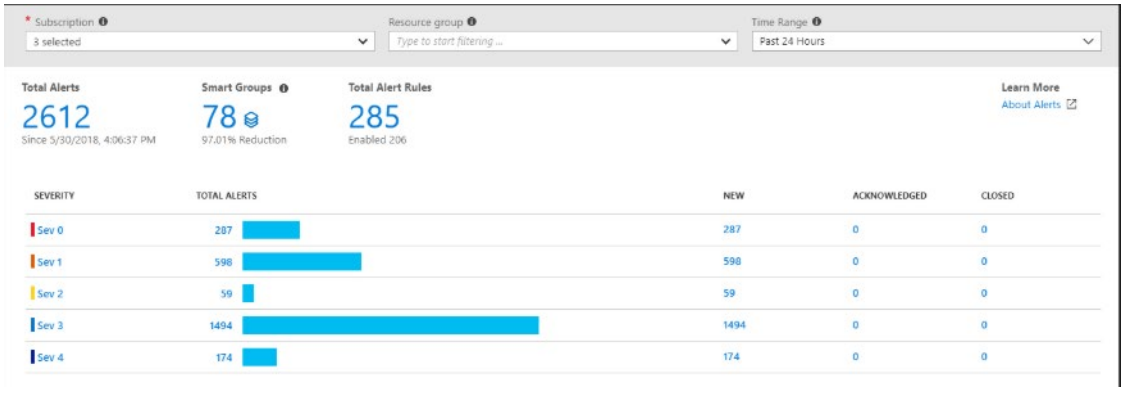
Smart Groups are in preview.

Smart groups are aggregations of alerts based on machine learning algorithms, which can help reduce alert noise and aid in trouble-shooting. **Learn more about Smart Groups**[22] and **how to manage your smart groups**[23].

Alerts experience

The default Alerts page provides a summary of alerts that are created within a particular time window. It displays the total alerts for each severity with columns that identify the total number of alerts in each state for each severity. Select any of the severities to open the **All Alerts**[24] page filtered by that severity.

It does not show or track older **classic alerts**[25]. You can change the subscriptions or filter parameters to update the page.



21 https://aka.ms/managing-alert-smart-group-states
22 https://aka.ms/smart-groups
23 https://aka.ms/managing-smart-groups
24 https://docs.microsoft.com/en-us/azure/azure-monitor/platform/alerts-overview#all-alerts-page
25 https://docs.microsoft.com/en-us/azure/azure-monitor/platform/alerts-overview#classic-alerts

You can filter this view by selecting values in the dropdown menus at the top of the page.

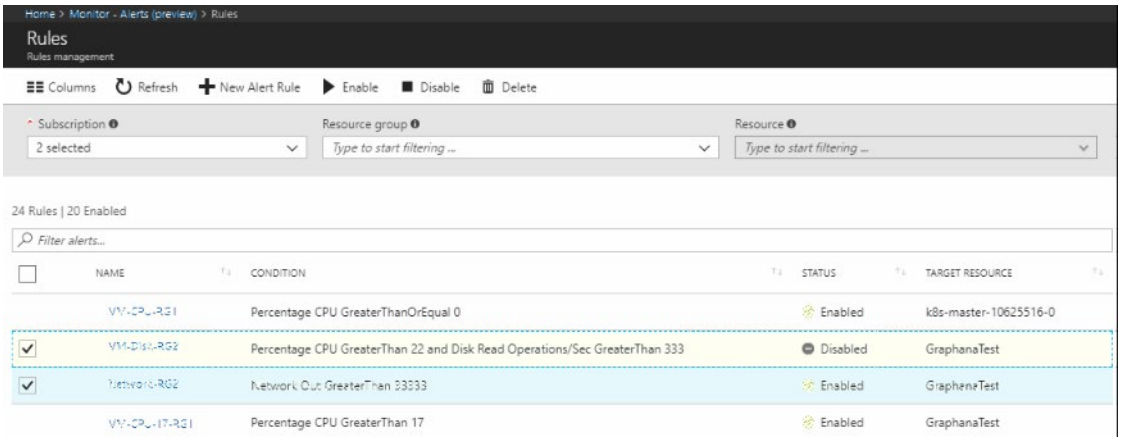
Column	Description
Subscription	Select up to five Azure subscriptions. Only alerts in the selected subscriptions are included in the view.
Resource group	Select a single resource group. Only alerts with targets in the selected resource group are included in the view.
Time range	Only alerts fired within the selected time window are included in the view. Supported values are the past hour, the past 24 hours, the past 7 days, and the past 30 days.

Select the following values at the top of the Alerts page to open another page.

Value	Description
Total alerts	The total number of alerts that match the selected criteria. Select this value to open the All Alerts view with no filter.
Smart groups	The total number of smart groups that were created from the alerts that match the selected criteria. Select this value to open the smart groups list in the All Alerts view.
Total alert rules	The total number of alert rules in the selected subscription and resource group. Select this value to open the Rules view filtered on the selected subscription and resource group.

Manage alert rules

Click on Manage alert rules to show the Rules page. Rules is a single place for managing all alert rules across your Azure subscriptions. It lists all alert rules and can be sorted based on target resources, resource groups, rule name, or status. Alert rules can also be edited, enabled, or disabled from this page.



Create an alert rule

Alerts can be authored in a consistent manner regardless of the monitoring service or signal type. All fired alerts and related details are available in single page.

You create a new alert rule with the following three steps:

1. Pick the *target* for the alert.
2. Select the *signal* from the available signals for the target.
3. Specify the *logic* to be applied to data from the signal.

This simplified authoring process no longer requires you to know the monitoring source or signals that are supported before selecting an Azure resource. The list of available signals is automatically filtered based on the target resource that you select. Also based on that target, you are guided through defining the logic of the alert rule automatically.

You can learn more about how to create alert rules in **Create, view, and manage alerts using Azure Monitor**[26].

Alerts are available across several Azure monitoring services. For information about how and when to use each of these services, see **Monitoring Azure applications and resources**[27]. The following table provides a listing of the types of alert rules that are available across Azure. It also lists what's currently supported in which alert experience.

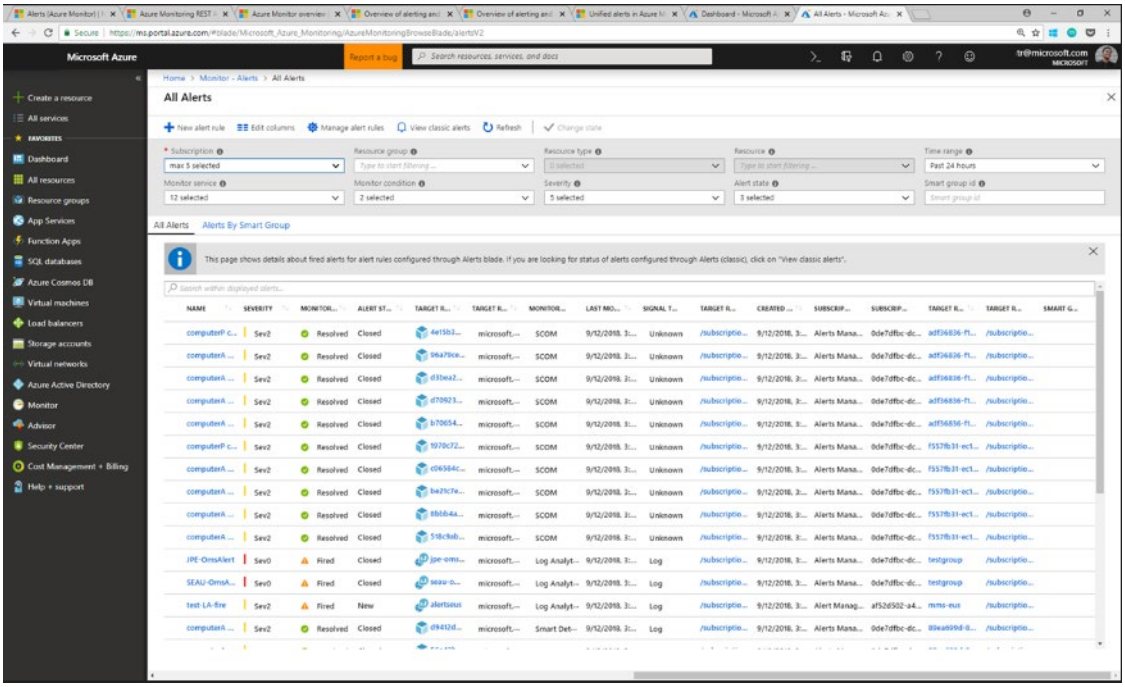
Previously, Azure Monitor, Application Insights, Log Analytics, and Service Health had separate alerting capabilities. Overtime, Azure improved and combined both the user interface and different methods of alerting. This consolidation is still in process. As a result, there are still some alerting capabilities not yet in the new alerts system.

Monitor source	Signal type	Description
Service health	Activity log	Not supported. See **Create activity log alerts on service notifications** *(https://docs.microsoft.com/en-us/azure/azure-monitor/platform/alerts-activity-log-service-notifications)*.
Application Insights	Web availability tests	Not supported. See **Web test alerts** *(https://docs.microsoft.com/en-us/azure/application-insights/app-insights-monitor-web-app-availability)*. Available to any website that's instrumented to send data to Application Insights. Receive a notification when availability or responsiveness of a website is below expectations.

26 https://docs.microsoft.com/en-us/azure/azure-monitor/platform/alerts-metric
27 https://docs.microsoft.com/en-us/azure/azure-monitor/overview

All alerts page

Click on Total Alerts to see the all alerts page. Here you can view a list of alerts that were created within the selected time window. You can view either a list of the individual alerts or a list of the smart groups that contain the alerts. Select the banner at the top of the page to toggle between views.



You can filter the view by selecting the following values in the dropdown menus at the top of the page.

Column	Description
Subscription	Select up to five Azure subscriptions. Only alerts in the selected subscriptions are included in the view.
Resource group	Select a single resource group. Only alerts with targets in the selected resource group are included in the view.
Resource type	Select one or more resource types. Only alerts with targets of the selected type are included in the view. This column is only available after a resource group has been specified.
Resource	Select a resource. Only alerts with that resource as a target are included in the view. This column is only available after a resource type has been specified.
Severity	Select an alert severity, or select All to include alerts of all severities.
Monitor condition	Select a monitor condition, or select All to include alerts of conditions.

Column	Description
Alert state	Select an alert state, or select All to include alerts of states.
Monitor service	Select a service, or select All to include all services. Only alerts created by rules that use service as a target are included.
Time range	Only alerts fired within the selected time window are included in the view. Supported values are the past hour, the past 24 hours, the past 7 days, and the past 30 days.

Select Columns at the top of the page to select which columns to display.

Blameless PostMortems and a Just Culture

A Blameless Post Mortem

Anyone who's worked with technology at any scale is familiar with failure. Failure cares not about the architecture designs you drudge over, the code you write and review, or the alerts and metrics you meticulously pore through. So: failure happens. This is a foregone conclusion when working with complex systems. But what about those failures that have resulted due to the actions (or lack of action, in some cases) of individuals? What do you do with those careless humans who caused everyone to have a bad day?

Maybe they should be fired. Or maybe they need to be prevented from touching the dangerous bits again. Or maybe they need more training.

This is the traditional view of "human error", which focuses on the characteristics of the individuals involved. This is called the "Bad Apple Theory" – get rid of the bad apples, and you'll get rid of the human error. Seems simple, right? Organizations that have pioneered DevOps are shying away from this traditional view. These DevOps practising organizations instead want to view mistakes, errors, slips, lapses, etc. with a perspective of *learning*. Having blameless Post-Mortems on outages and accidents are part of that.

A Blameless Post-Mortem

What does it mean to have a 'blameless' Post-Mortem?

Does it mean everyone gets off the hook for making mistakes? No.

Well, maybe. It depends on what "gets off the hook" means. Let me explain.

Having a **Just Culture** means that you're making effort to balance safety ***and*** accountability. It means that by investigating mistakes in a way that focuses on the situational aspects of a failure's mechanism and the decision-making process of individuals proximate to the failure, an organization can come out safer than it would normally be if it had simply punished the actors involved as a remediation.

Having a "blameless" Post-Mortem process means that engineers whose actions have contributed to an accident can give a detailed account of:

- what actions they took at what time,
- what effects they observed,
- expectations they had,
- assumptions they had made,
- and their understanding of timeline of events as they occurred.

...and that they can give this detailed account ***without fear of punishment or retribution.***

Why shouldn't they be punished or reprimanded? Because an engineer who thinks they're going to be reprimanded are *disincentivized* to give the details necessary to get an understanding of the mechanism, pathology, and operation of the failure. This lack of understanding of how the accident occurred all but guarantees that it ***will*** repeat. If not with the original engineer, another one in the future.

If we go with "blame" as the predominant approach, then we're implicitly accepting that *deterrence* is how organizations become safer. This is founded in the belief that individuals, not situations, cause errors. It's also aligned with the idea there has to be some fear that **not** doing one's job correctly could lead to punishment. Because the fear of punishment will motivate people to act correctly in the future. Right?

This cycle of name/blame/shame can be looked at like this:

1. Engineer takes action and contributes to a failure or incident.
2. Engineer is punished, shamed, blamed, or retrained.
3. Reduced trust between engineers on the ground (the "sharp end") and management (the "blunt end") looking for someone to scapegoat
4. Engineers become silent on details about actions/situations/observations, resulting in "Cover-Your-Mistake" engineering (from fear of punishment)
5. Management becomes less aware and informed on how work is being performed day to day, and engineers become less educated on lurking or latent conditions for failure due to silence mentioned in #4, above
6. Errors more likely, latent conditions can't be identified due to #5, above
7. Repeat from step 1

We need to avoid this cycle. We want the engineer who has made an error give details about why (either explicitly or implicitly) he or she did what they did; why the action made sense to them at the time. This is paramount to understanding the pathology of the failure. The action made sense to the person at the time they took it, because if it hadn't made sense to them at the time, they ***wouldn't have taken the action in the first place.***

The base fundamental here is something **Erik Hollnagel**[28] has said:

We must strive to understand that accidents don't happen because people gamble and lose.
Accidents happen because the person believes that:
...what is about to happen is not possible,
...or what is about to happen has no connection to what they are doing,
...or that the possibility of getting the intended outcome is well worth whatever risk there is.

Allowing Engineers to Own Their Own Stories

A funny thing happens when engineers make mistakes and feel safe when giving details about it: they are not only willing to be held accountable, they are also enthusiastic in helping the rest of the company avoid the same error in the future. They are, after all, the most expert in their own error. They ought to be heavily involved in coming up with remediation items.

So technically, engineers are not at all "off the hook" with a blameless PostMortem process. They are very much on the hook for helping become safer and more resilient, in the end. And lo and behold: most engineers I know find this idea of making things better for others a worthwhile exercise.

So what do we do to enable a "Just Culture"?

- Encourage learning by having these blameless Post-Mortems on outages and accidents.
- The goal is to understand **how **an accident could have happened, in order to better equip ourselves from it happening in the future
- Gather details from multiple perspectives on failures, and don't punish people for making mistakes.
- Instead of punishing engineers, we instead give them the requisite authority to improve safety by allowing them to give detailed accounts of their contributions to failures.
- Enable and encourage people who *do* make mistakes to be the experts on educating the rest of the organization how not to make them in the future.

28 http://www.erikhollnagel.com/

- Accept that there is always a discretionary space where humans can decide to make actions or not, and that the judgement of those decisions lie in hindsight.
- Accept that the **Hindsight Bias**[29] will continue to cloud our assessment of past events, and work hard to eliminate it.
- Accept that the **Fundamental Attribution Error**[30] is also difficult to escape, so we focus on the environment and circumstances people are working in when investigating accidents.
- Strive to make sure that the blunt end of the organization understands how work is actually getting done (as opposed to how they imagine it's getting done, via Gantt charts and procedures) on the sharp end.
- The sharp end is relied upon to inform the organization where the line is between appropriate and inappropriate behavior. This isn't something that the blunt end can come up with on its own.

Failure happens. In order to understand how failures happen, we first have to understand our ***reactions*** to failure.

One option is to assume the single cause is incompetence and scream at engineers to make them "pay attention!" or "be more careful!"

Another option is to take a hard look at how the accident actually happened, treat the engineers involved with respect, and *learn* from the event.

- **https://blogs.msdn.microsoft.com/bharry/2018/03/02/a-good-incident-postmortem/**
- **https://stories.visualstudio.com/devops/**

29 http://en.wikipedia.org/wiki/Hindsight
30 http://en.wikipedia.org/wiki/Fundamental_attribution_error

Module 4 Course Conclusion

Final Exam

Final Exam Questions

DevOps_ICF.101_CB

Which of hte statements about feature flags are true? (Select three.)

- ☐ Feature flags let you decouple releases from feature deployment.
- ☐ Feature flags enable enhanced feedback loops with customers.
- ☐ Feature flags don't need to be removed from the code after the feature has been released.
- ☐ Releases with substantially mitigated risk during rollback of a feature.

DevOps_ICF.102_MC

Is this statement correct?
Azure DevOps allows integration with Microsoft Teams to improve the visibility and collaboration within the software development team.

- ☐ true
- ☐ false

DevOps_ICF.103_CB

From the options below select all the advantages you are expected to get by using in app feedback in your products.

- ☐ Context sensitive feedback
- ☐ A channel of feedback always available to users
- ☐ A place for customer to see all product updates
- ☐ A please for customers to see product feature roadmap

DevOps_ICF.104_CB

Which of the release gates are available within Azure Pipelines? (Select three.)

- ☐ Query Azure Monitor Alerts
- ☐ Query Work Items
- ☐ Servicenow change management
- ☐ Build monitoring

DevOps_ICF.105_CB

Select all the features available to you when using Application Insights for Application Performance Monitoring (APM). (Select three.)

- ☐ Auto fix code issues
- ☐ Application Map
- ☐ Profiler
- ☐ Live metrics stream

DevOps_ICF.106_CB

Which IDE does App Insights integrate with to provide developers application performance monitoring insights from deployed application environments? (Select two.)

- ☐ Visual Studio
- ☐ Visual Studio Code
- ☐ Eclipse
- ☐ PyCharm

DevOps_ICF.107_CB

Which of the dashboarding solutions does App Insights provide integration with?

- ☐ PowerBi
- ☐ Tableau
- ☐ Grafana
- ☐ Azure Monitor

DevOps_ICF.108_CB

When defining Alerts in Azure Monitor which action groups are available to you? (Select three.)

- ☐ Voicemail
- ☐ Automation Runbook
- ☐ Webhook
- ☐ ITSM

DevOps_ICF.109_MC

Is the following statement correct?

App Insights allows integration with Azure Boards using work items?

- ☐ true
- ☐ false

DevOps_ICF.110_CB

Which of the following management solutions are supported in Azure Monitor? (Select three.)

- ☐ Azure Activity
- ☐ Change Management
- ☐ Auto delete
- ☐ SQL Assessment

DevOps_ICF.111_CB

Which of the following statements are correct? (select two.)

- ☐ Service Map automatically discovers application components on Windows and Linux systems and maps the communication between services.
- ☐ Service Map shows connections between servers, processes, inbound and outbound connection latency, and ports.
- ☐ Service Map shows the services available in Azure
- ☐ Service Map automatically discovers application components on Windows, Linux and Mac systems and maps the communication between services.

DevOps_ICF.112_MC

Is this statement correct?

Azure Monitor provides monitoring solution for Azure Virtual Machines.

- ☐ true
- ☐ false

DevOps_ICF.113_CB

Azure App Insights provides the following officially supported SDKs? (Select three.)

- ☐ JavaScript
- ☐ Asp.Net Core
- ☐ Python
- ☐ Node.js

DevOps_ICF.114_CB

Select the IT Service Management connectors supported by Azure Monitor. (Select three.)

- ☐ ServiceNow
- ☐ System Center Service Manager
- ☐ Provance
- ☐ Charm well

DevOps_ICF.115_MC

Is this statement correct?

Azure App Insights supports Debug snapshots sampled from live operations, with parameter values.

- ☐ true
- ☐ false

DevOps_ICF.116_CB

Which of the following features are supported by analytics in Visual Studio App Center?

- ☐ Crash and Error Analytics
- ☐ Grouping
- ☐ Attachments
- ☐ Events before a crash

DevOps_ICF.117_CB

Which bug trackers are supported by Visual Studio App Center? (Select three.)

- ☐ Azure Boards
- ☐ GitHub
- ☐ Jira
- ☐ Trello

DevOps_ICF.118_CB

Which are following are techniques to capture customer satisfaction? (Select three.)

- ☐ Customer Satisfaction Score (CSAT)
- ☐ Customer Effort Score (CES)
- ☐ Azure Consumption Score (ACS)
- ☐ Net Promoter Score® (NPS)

DevOps_ICF.119_MC

Which of the following DevOps practice enables a continuous improvement culture?

- ☐ Blameless postmortems of incidents
- ☐ Blame and fire individuals associated with an incident
- ☐ Ignore incidents as they can always happen in live site
- ☐ Only investigate incidents that have at least repeated 10 times in production

DevOps_ICF.120_CB

Which of the following statements are correct about a site reliability engineer? (Select three.)

- ☐ Proactively monitor and review application performance
- ☐ Create and maintain operational runbooks
- ☐ Ensure software has good logging and diagnostics
- ☐ Leave critical issues to development team to investigate

Answers

DevOps_ICF.101_CB

Which of hte statements about feature flags are true? (Select three.)

- ■ Feature flags let you decouple releases from feature deployment.
- ■ Feature flags enable enhanced feedback loops with customers.
- ☐ Feature flags don't need to be removed from the code after the feature has been released.
- ■ Releases with substantially mitigated risk during rollback of a feature.

DevOps_ICF.102_MC

Is this statement correct?
Azure DevOps allows integration with Microsoft Teams to improve the visibility and collaboration within the software development team.

- ■ true
- ☐ false

DevOps_ICF.103_CB

From the options below select all the advantages you are expected to get by using in app feedback in your products.

- ■ Context sensitive feedback
- ■ A channel of feedback always available to users
- ☐ A place for customer to see all product updates
- ☐ A please for customers to see product feature roadmap

DevOps_ICF.104_CB

Which of the release gates are available within Azure Pipelines? (Select three.)

- ■ Query Azure Monitor Alerts
- ■ Query Work Items
- ■ Servicenow change management
- ☐ Build monitoring

DevOps_ICF.105_CB

Select all the features available to you when using Application Insights for Application Performance Monitoring (APM). (Select three.)

- ☐ Auto fix code issues
- ■ Application Map
- ■ Profiler
- ■ Live metrics stream

DevOps_ICF.106_CB

Which IDE does App Insights integrate with to provide developers application performance monitoring insights from deployed application environments? (Select two.)

- ■ Visual Studio
- ■ Visual Studio Code
- ☐ Eclipse
- ☐ PyCharm

DevOps_ICF.107_CB

Which of the dashboarding solutions does App Insights provide integration with?

- ■ PowerBi
- ☐ Tableau
- ■ Grafana
- ■ Azure Monitor

DevOps_ICF.108_CB

When defining Alerts in Azure Monitor which action groups are available to you? (Select three.)

- ☐ Voicemail
- ■ Automation Runbook
- ■ Webhook
- ■ ITSM

DevOps_ICF.109_MC

Is the following statement correct?

App Insights allows integration with Azure Boards using work items?

- ■ true
- ☐ false

DevOps_ICF.110_CB

Which of the following management solutions are supported in Azure Monitor? (Select three.)

- ■ Azure Activity
- ■ Change Management
- ☐ Auto delete
- ■ SQL Assessment

DevOps_ICF.111_CB

Which of the following statements are correct? (select two.)

- ■ Service Map automatically discovers application components on Windows and Linux systems and maps the communication between services.
- ■ Service Map shows connections between servers, processes, inbound and outbound connection latency, and ports.
- ☐ Service Map shows the services available in Azure
- ☐ Service Map automatically discovers application components on Windows, Linux and Mac systems and maps the communication between services.

DevOps_ICF.112_MC

Is this statement correct?

Azure Monitor provides monitoring solution for Azure Virtual Machines.

- ■ true
- ☐ false

DevOps_ICF.113_CB

Azure App Insights provides the following officially supported SDKs? (Select three.)

- ■ JavaScript
- ■ Asp.Net Core
- ☐ Python
- ■ Node.js

DevOps_ICF.114_CB

Select the IT Service Management connectors supported by Azure Monitor. (Select three.)

- ■ ServiceNow
- ■ System Center Service Manager
- ■ Provance
- ☐ Charm well

DevOps_ICF.115_MC

Is this statement correct?

Azure App Insights supports Debug snapshots sampled from live operations, with parameter values.

- ■ true
- ☐ false

DevOps_ICF.116_CB

Which of the following features are supported by analytics in Visual Studio App Center?

- ■ Crash and Error Analytics
- ■ Grouping
- ■ Attachments
- ■ Events before a crash

DevOps_ICF.117_CB

Which bug trackers are supported by Visual Studio App Center? (Select three.)

- ■ Azure Boards
- ■ GitHub
- ■ Jira
- ☐ Trello

DevOps_ICF.118_CB

Which are following are techniques to capture customer satisfaction? (Select three.)

- ■ Customer Satisfaction Score (CSAT)
- ■ Customer Effort Score (CES)
- ☐ Azure Consumption Score (ACS)
- ■ Net Promoter Score® (NPS)

DevOps_ICF.119_MC

Which of the following DevOps practice enables a continuous improvement culture?

- ■ Blameless postmortems of incidents
- ☐ Blame and fire individuals associated with an incident
- ☐ Ignore incidents as they can always happen in live site
- ☐ Only investigate incidents that have at least repeated 10 times in production

DevOps_ICF.120_CB

Which of the following statements are correct about a site reliability engineer? (Select three.)

- ■ Proactively monitor and review application performance
- ■ Create and maintain operational runbooks
- ■ Ensure software has good logging and diagnostics
- ☐ Leave critical issues to development team to investigate