

AZ 400

part 1



Outline

- 1. Analyzing Metrics**
- 2. Designing and Implementing a Source Control Strategy**
- 3. Planning and Implementing Branching Strategies for the Source Code**
- 4. Configuring Repositories**

Analyzing Metrics

Big Picture: What Is Site Reliability Engineering (SRE)?



What Is Reliability?

The application can be used when
and how it is expected to

- Availability
- Performance
- Latency
- Security

Measuring Reliability

How to determine an acceptable uptime

SLA

**Service Level
Agreement**

The agreement made with customers about how available the service is

SLO

**Service Level
Objective**

The goals that the service wants to reach to meet the agreement

SLI

**Service Level
Indicator**

Actual service metrics behind the commitments

Why Do We Need Site Reliability Engineering?



Ownership

Generally, not focused on by the teams best-suited to handle the issues



Work Silos

Software engineers would develop the code and then hand it off to deploy and maintain



Full Stack

Code can be edited to increase performance and develop automation to reduce errors

Key Concepts

Key Concepts



Feedback

Feedback loops designed for continuous improvement



Automation

Automate as much as possible



Measure Everything

Understand by collecting data on how everything is working



Small Changes

Roll out smaller changes — but more often



Alerting

Use the data to create alerts on



Risk

Embrace risk as a part of the process and blamelessly learn from mistakes

Reliability

Doesn't matter how many features something has if the product doesn't work

SRE vs DevOps



Key Concepts

- ✓ Reliability can be described in availability, performance, latency, and security.
- ✓ SRE takes development practices to solve operations problems.
- ✓ Site Reliability Engineers are effective because they can edit the code across the stack.

Key Concepts

- ✓ A Service Level Agreement (SLA) is the agreement made with the customer.
- ✓ Service Level Objectives (SLO) are the goals.
- ✓ Service Level Indicators (SLI) are the metrics.

Key Concepts

- ✓ Feedback, Measure Everything, Alerting, Automation, Small Changes, and Embrace Risk are all key SRE concepts.
- ✓ SRE focuses on production.
- ✓ DevOps focuses on deployments.

Azure Monitor



Azure Monitor

Central location to gather information from your applications, infrastructure, and Azure resources

METRICS

LOGS

Data that measure how
the system is performing
at a certain time

Messages about certain
events in a system

Azure Monitor



Azure Monitor

All services > Monitor

Monitor | Metrics

Microsoft

Search (Cmd+/)

Overview

Activity log

Alerts

Metrics

Logs

Service Health

Workbooks

Insights

Applications

Virtual Machines

Storage accounts

Containers

Networks

SQL (preview)

New chart Refresh Share Feedback

Local Time: Last 24 hours (Automatic - 5 min...)

Avg Percentage CPU for casvmss2j

Add metric Add filter Apply splitting

Line chart Drill into Logs New alert rule Pin to dashboard

Scope: casvmss2j Metric Namespace: Virtual Machine Host Metric: Percentage CPU Aggregation: Avg

Count

Avg

Min

Max

Sum

60%

50%

40%

30%

20%

10%

0%

Wed 07

6 AM

12 PM

6 PM

UTC-04:00

Azure Monitor

CASweets Dashboard

Shared dashboard

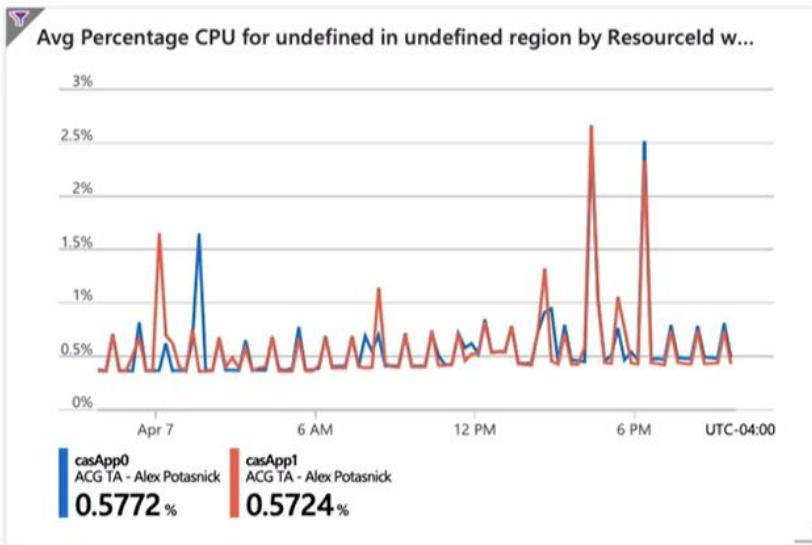
+ New dashboard Refresh Full screen | Edit Manage sharing Download Clone Assign tags Delete | Feedback

Auto refresh : Off

UTC Time : Past 24 hours

+ Add filter

Last updated: a few seconds ago



Azure Monitor

- ✓ Azure Monitor collects metrics and logs from applications, infrastructure, and Azure resources.
- ✓ Create charts in Metrics Explorer by selecting a scope, namespace, metric, aggregation, and time period.

Implementing Application Health Checks

Implementing Application Health Checks

Our Scenario

What Is Application Insights?

Availability

Demo

- Configure URL Ping Test
- Configure Health Alert

Our Scenario

Ctrl Alt Sweets

How is the team handling this?

- ✓ Using Azure Monitor Metrics for infrastructure health
- ✓ Manual checks for application health
- ✓ Looking for ways to automate



Availability

What Is Application Insights?

Specifically designed to help you understand the health and usage of your application

How is the application doing and how is it being used?



Application exceptions



Application dependencies



Performance information

Availability



URL Ping Test

Send an HTTP request to a specific URL to test availability

- Lets you know the request duration and success rate
- Dependent requests test availability of files on a webpage
- Enable retries is recommended
- Test frequency
- Minimum of 5 and maximum of 16 test locations
- Can create alerts

Create a test

Microsoft Azure Search resources, services, and docs (G+) Home Monitor appi-2juwmhr22al4i

appi-2juwmhr22al4i | Availability Add test Refresh View in Logs SLA Report Cards

Click on 'SLA Report' button above to try our new SLA Report for downtime.

Local Time: Last 24 hours Filter

Availability ?

100.00%
80.00%
60.00%
40.00%
20.00%
0.00%

06:00 PM Fri 9 06:00 AM 08:00 AM

02:34 PM 06:00 PM Fri 9 06:00 AM

Select availability test Search to filter items...

Create test

Learn more how to setup

Test type URL ping test

URL * https://wapp-2juwmhr22al4i.azurewebsites.net

Parse dependent requests ?

Enable retries for availability test failures ?

Test frequency 5 minutes

Test locations 5 location(s) configured ?

Success criteria HTTP response: 200, Test Timeout: 120 seconds

Alerts Enabled

Create an action group

Microsoft Azure Search resources, services, and docs (G+) ✉️ 📱 📡 🚧 🛠️ ? 😊 👤

Home > Monitor > appi-2juwmhr22al4i > Rules > test1-appi-2juwmhr22al4i >

Create action group

Basics Notifications Actions Tags Review + create

Notifications

Configure the method in which users will be notified when the action group triggers. Select notification types, receiver details and add a unique description. This step is optional.

Notification type	Name	Selected
Email/SMS message/Push/Voice	tim	edit

Please configure the notification by clicking the edit button.

Email
Email edit

SMS (Carrier charges may apply)
Country code 1

Azure app Push Notifications
Azure account email edit

Voice
Country code 1

Phone number

Failing availability test

Microsoft Azure Search resources, services, and docs (G+ /) 5

Home > Monitor > appi-2juwmhr22al4i

appi-2juwmhr22al4i | Availability

Application Insights

Search (Cmd+/) Add test Refresh View in Logs SLA Report Copy link Troubleshoot Feedback

Click on 'SLA Report' button above to try our new SLA Report for downtime and outages analysis! Dismiss

Overview Activity log Access control (IAM) Tags Diagnose and solve problems

Investigate Application map Smart Detection Live metrics Transaction search Availability Failures Performance

06:00 PM Fri 9 06:00 AM 12:00 PM 02:50 PM 02:50 PM

Select availability test Search to filter items...

AVAILABILITY TEST	20 MIN	AVAILABIL...	DURATION (AVG)
Overall	57.14%	57.14%	599 ms
test1	57.14%	57.14%	599 ms
Australia East	66.67%	66.67%	1.17 sec
Brazil South	50.00%	50.00%	833 ms
Central US	60.00%	60.00%	275 ms
East Asia	50.00%	50.00%	826 ms
East US	60.00%	60.00%	212 ms

Implementing Application Health Checks

- ✓ Application Insights provides informational data on your application such as how it is doing and how it is being used.
- ✓ URL ping tests send an HTTP request to your URL to see if there is a response.

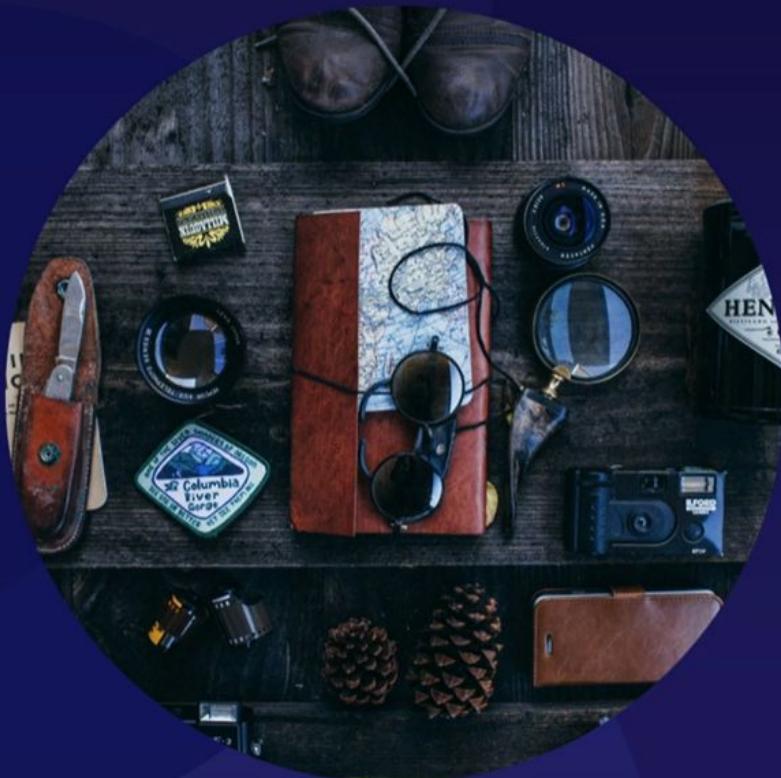
Implementing Application Health Checks

- ✓ URL ping tests can be found in the Availability section of Application Insights.
- ✓ Dependent requests test the availability of files on a webpage.
- ✓ Microsoft recommends enabling retries on the tests.

Implementing Application Health Checks

- ✓ Once a test has been created, you can configure alerts to be sent on failures.
- ✓ Create Action Groups to tell the alert who to notify and how.

Exploring System Load and Failure Conditions



Everything fails sometimes

Whether it's hardware or software, it will eventually fail at some point.

The goal is to be prepared for any eventuality before being notified by the end user.

Exploring System Load and Failure Conditions



Find Single Points of Failure

- Fault points: any place in the architecture that can fail
- Fault modes: all the ways a fault point can fail

Rate Risk and Severity

How likely it will fail and what is the impact

Determine Response

How the application will respond and recover from a failure

Exploring System Load and Failure Conditions



Understand the Application

Understand what and how the application works.



Criticality

Determine if the system is critical or noncritical.



Dependencies

Know all the connected components.

Exploring System Load and Failure Conditions

How Can We Reduce Failures?

✓ Fault Domains

Implement fault domains where applicable.

✓ Cross-Region

Use geo-redundant storage and have a read-access data replica and site recovery plan in another region.

✓ Zones

Use zone-redundant storage, data, and availability zones where applicable.

✓ Scaling

Implement autoscaling.

Exploring System Load and Failure Conditions

Performance Testing

1 Load Testing

Test the application can handle normal to heavy load.

2 Stress Testing

Attempt to overload the system.

3 Spikes

A best practice is to include a buffer to account for random spikes.



Understanding Baseline Metrics

Understanding Baseline Metrics

Our Scenario

Why Create a Baseline?

How Do We Create a Baseline?

Demo

- Explore Azure Monitor Insights

Our Scenario

Ctrl Alt Sweets

How is the team handling this?

- ✓ Configuration drift in Testing environment from Production
- ✓ “Normal” system behavior is different for each environment
- ✓ Difficult to do performance testing without a proper baseline



Tom
TEST/QA

Team Lead
TEST/QA

Understanding Baseline Metrics



Why create a baseline?

Once we have a baseline, we can understand what a healthy state is.

Any deviations from the baseline can be alerted on and can also be used as starting points for improvements.

Understanding Baseline Metrics

How do we create a baseline?

Azure provides various tools that can be used to understand what the baseline metrics and workloads are.

Log Analytics and Metrics Explorer

Create queries and charts to capture and analyze data

Azure Monitor Insights

Provides recommended metrics and dashboards for several services

Application Insights

Provides recommended metrics and dashboards for an application

Understanding Baseline Metrics

Microsoft Azure Search resources, services, and docs (G+) Home Monitor Insights ... X

Resource Group Monitoring Azure Monitor Run Diagnostics Refresh Provide Feedback

Performance Map Health (preview)

Time range: **Last 30 minutes as of 23 Apr 10:05**

View Workbooks Legend

The map displays a central node labeled "casApp0" with the following details:

- 29 Processes
- 33 Clients

Connections are shown to four other nodes, each representing a server or port:

- Port: 443 → 6 Servers
- Port: 53 → 1 Servers
- Port: 80 → 3 Servers

casApp0 Machine Summary

Quick links

- Connection details

Fully Qualified Domain Name: casApp0

Operating System: Windows Server 2019 Version 1803 build 17763

IPv4 Addresses: 10.1.1.4/24

Health

Machine properties

Azure VM properties

Properties Log Events Alerts Connections Overview Changes

Understanding Baseline Metrics

Microsoft Azure Search resources, services, and docs (G+) ...

Home > Monitor >

Insights

Resource Group Monitoring Azure Monitor Run Diagnostics Refresh Provide Feedback

Performance Map Health (preview)

Time range: Last hour as of 23 Apr 10:05 View Workbooks

Logical Disk Performance

DISK	CURRENT SIZE (...	CURRENT USED ...	P95 IOPs RE...	P95 IOPs WR...	P95 IOPs TO...	P95 MB/s RE...	P95 MB/s W...	P95 MB/s TO...	P95 LATENCY READ (...	P95 LATENCY WRITE ...	P9...
C:	126.51	11%	0.34	5.43	5.53	0.1	0.14	0.24	0.53	0.88	0.8...
D:	100	1%	0	0	0	0	0	0	0	0	0...
Total	226.51	7%	0.34	5.43	5.53	0.1	0.14	0.24	0.53	1.4	1.3...

CPU Utilization %

Avg	Min	50th	90th	95th	Max	...
100%						
80%						
60%						
40%						

Available Memory

Avg	Max	50th	10th	5th	Min	...
5.6GB						
4.7GB						
3.7GB						
2.8GB						

Properties Log Events Alerts Changes

Understanding Baseline Metrics

The screenshot shows the Microsoft Azure Monitor - cassa Insights page. The top navigation bar includes the Microsoft Azure logo, a search bar, and various icons for account management. The left sidebar lists navigation links such as Home, Monitor, cassa, Overview, Activity log, Tags, Diagnose and solve problems, Access Control (IAM), Data migration, Events, Storage Explorer (preview), Settings, Access keys, Geo-replication, CORS, and Configuration. The main content area displays the cassa Insights page with tabs for Overview, Failures, Performance, Availability, and Capacity. The Overview tab is selected, showing a summary of storage metrics: Availability (100%), Transactions (25), Success E2E Latency (24.28 ms), and Success Server Latency (24.12 ms). Below these metrics, there are sections for 'Transactions by storage type' and 'Transactions by API name'. The overall interface is clean and modern, typical of the Azure portal.

Understanding Baseline Metrics

Microsoft Azure Search resources, services, and docs (G+/-) ✉️ 🔍 🚙 🌐 🛡️ ? 😊 👤

Home > Monitor >

appi-2juwmhr22al4i ✖️ ⚙️ ⋮

Application Insights

Search (Cmd +/)

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Investigate

Application map

Smart Detection

Live metrics

Transaction search

Availability

Failures

Performance

Troubleshooting guides (preview)

Application Dashboard Getting started Search Logs Monitor resource group Feedback Favorites Rename ⋮

Failed requests (Count)
appi-2juwmhr22al4i
0

9:15 AM 9:30 AM 9:45 AM 10 AM UTC-04:00

Server response time (Avg)
appi-2juwmhr22al4i
--

9:15 AM 9:30 AM 9:45 AM 10 AM UTC-04:00

Server requests

100
80
60
40
20
0

9:15 AM 9:30 AM 9:45 AM 10 AM UTC-04:00

Availability

100%
80%
60%
40%
20%
0%

9:15 AM 9:30 AM 9:45 AM 10 AM UTC-04:00

Understanding Baseline Metrics

Microsoft Azure Search resources, services, and docs (G+ /) More

appi-2juwmhr22al4i Dashboard

Shared dashboard

New dashboard Refresh Full screen Edit Manage sharing Download Clone Assign tags Delete Feedback

Auto refresh : Off UTC Time : Past 24 hours Add filter Last updated: 41 minutes ago

appi-2juwmhr22al4i Application Insights ... Smart Detection appi-2juwmhr2... Live Stream 1 Servers -- App map APPI-2JU...

Usage

Edit 1 Users

Unique sessions and users

Sessions (Unique) appi-2juwmhr22al4i 1

Users (Unique) appi-2juwmhr22al4i 0

Reliability

Edit Failures appi-2juwmhr2...

Failed requests

Failed requests (Count) appi-2juwmhr22al4i 0

Responsiveness

Edit Perform... appi-2juwmhr2...

Server response time

Server response time (Avg) appi-2juwmhr22al4i --

Browser

Average page load time

Page load network co... appi-2juwmhr22al4i 1/2 --

Understanding Baseline Metrics

- ✓ A baseline can help you identify when a system is not in a healthy state so that alerts and improvements can be implemented.
- ✓ Azure Monitor Insights provides recommended charts and metrics for Azure resources.
- ✓ Application Insights provides recommended charts and metrics for applications.

Discovering Application Insights Smart Detection and Dynamic Thresholds

Discovering Application Insights Smart Detection and Dynamic Thresholds

Our Scenario

Dynamic Threshold Advantages

Application Insights Smart Detection

Smart Detection Categories

Demo

- Create an Alert with Dynamic Thresholds
- Create Smart Detection Alerts

Our Scenario

Ctrl Alt Sweets

How is the team handling this?

- ✓ Now using baselines for their performance testing
- ✓ Looking into using those health baselines to create alerts
- ✓ Want alerts to be adaptable to future changes that might alter the baseline



Dynamic Thresholds

3 Dynamic Threshold Alert Advantages

1

Machine Learning

Machine learning is applied to analyze historical data to understand when there are anomalies.

2

Less Manual Work

Don't have to manually figure out what the thresholds should be.

3

Set It and Forget It

Can be set to apply to any future resources and will continue to analyze data and adapt to changes.

Application Insights Smart Detection

Application Insights Smart Detection

1 Machine Learning

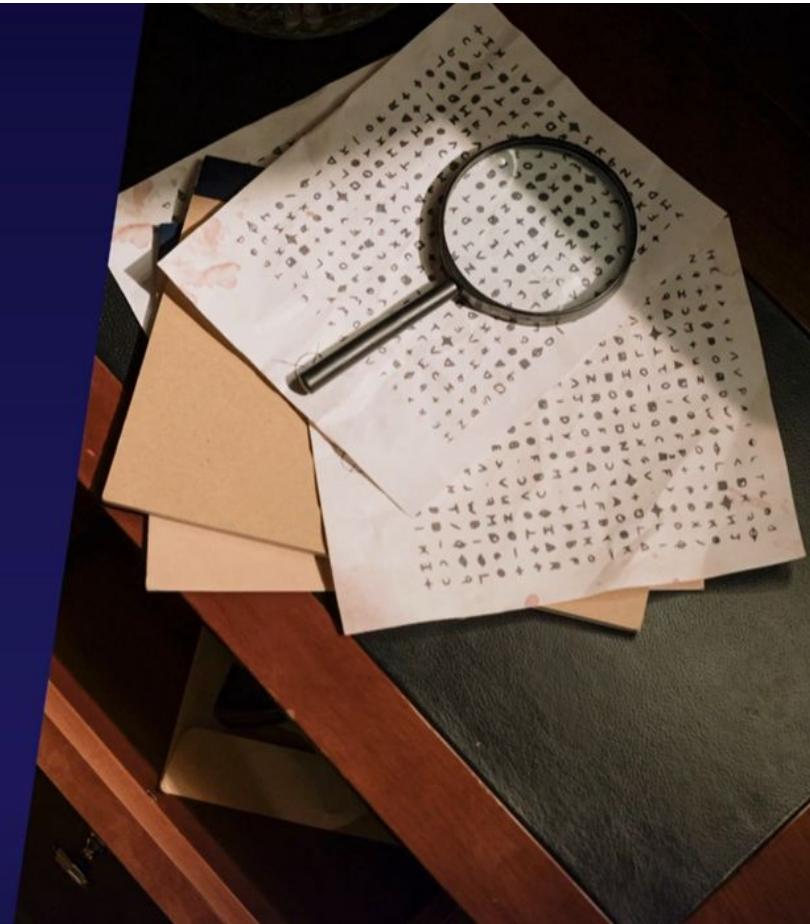
Analyze telemetry data to detect anomalies.

2 Built-In

Once there is enough time and data to analyze, it will be configured automatically.

3 Alerting

Provides information based on findings as well as information as to why there might be an issue.



Smart Detection categories

Categories

- **Failures category:** Needs a minimum amount of data and 24 hours to start
- **Performance category:** Needs a minimum amount of data and 8 days to start



Failures

Failure Rates

Figures out what the expected **number** of failures should be

Continuous Monitoring

Alerts in near real time

Alert Context

Provides information as to why it might have failed



Performance

Page Responses

If page takes too long to load or if operations or responses from dependencies are too slow

Daily Analysis

Sends a notification once a day

Alert Context

Provides information as to why it is running slow

Discovering Application Insights Smart Detection and Dynamic Thresholds

Home > Monitor >

Create alert rule

...

Create an alert rule to identify and address issues when important
When defining the alert rule, check that your inputs do not contain

Scope

Select the target resource you wish to monitor.

Resource

 casvmss2j

[Edit resource](#)

Condition

Configure when the alert rule should trigger by selecting a signal a

Condition name

No condition selected yet

[Add condition](#)

Configure signal logic

X

Dimension name	Operator	Dimension values	
Select dimension	=	0 selected	Add custom value

Alert logic

Threshold ⓘ

Static Dynamic

 Monitoring 1 time series (\$0.2/time series)

Operator ⓘ

Greater or Less than

Aggregation type * ⓘ

Average

  How does it work?

Threshold Sensitivity * ⓘ

Medium

Condition preview

Whenever the average percentage cpu is greater or less than dynamic thresholds for at least 4 times in the last 4 hours (4 aggregated points)

Evaluated based on

Aggregation granularity (Period) * ⓘ

1 hour

Frequency of evaluation ⓘ

Every 5 Minutes

▼ Advanced settings

Discovering Application Insights Smart Detection and Dynamic Thresholds

Home > Monitor >

Create alert rule

...

Create an alert rule to identify and address issues when important
When defining the alert rule, check that your inputs do not contain

Scope

Select the target resource you wish to monitor.

Resource

 casvmss2j

[Edit resource](#)

Condition

Configure when the alert rule should trigger by selecting a signal a

Condition name

No condition selected yet

[Add condition](#)

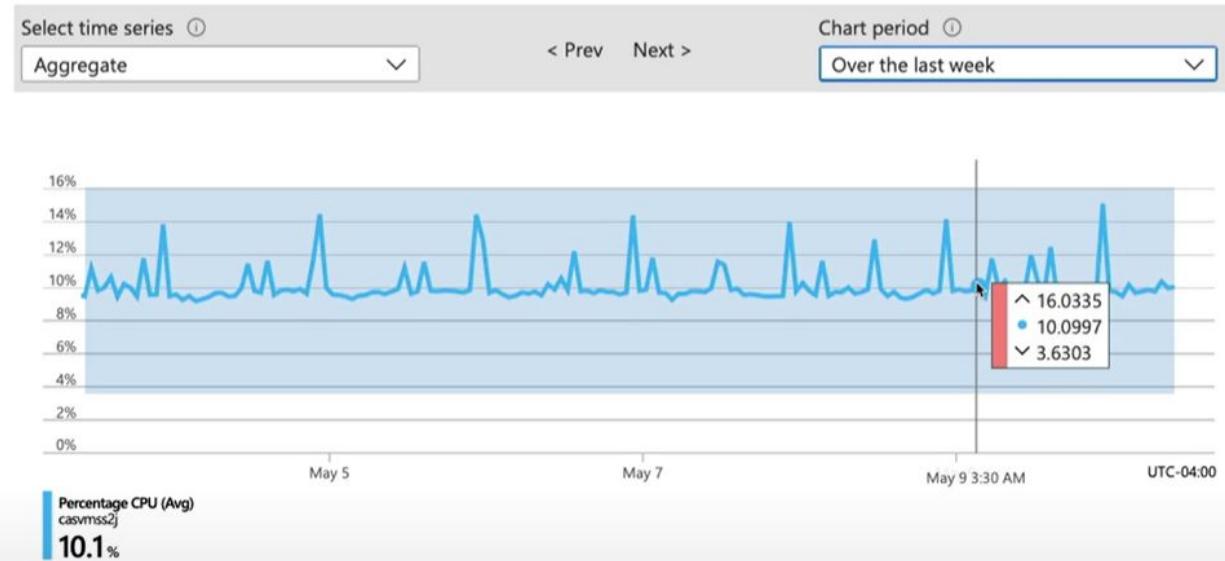
Configure signal logic

X

Beside the logic for triggering an alert, use the chart to view activity in the data.

[← Edit signal](#)

Selected signal: Percentage CPU (Platform)



Discovering Application Insights Smart Detection and Dynamic Thresholds

Home > Monitor >

Home > Monitor >

Create alert rule

Create an alert rule to identify and address issues when important
When defining the alert rule, check that your inputs do not contain

Scope

Select the target resource you wish to monitor.

Resource

 casvmss2j

[Edit resource](#)

Condition

Configure when the alert rule should trigger by selecting a signal a

Condition name

No condition selected yet

[Add condition](#)

Configure signal logic

Configure signal logic

Condition preview

Whenever the average percentage cpu is greater or less than dynamic thresholds for at least 4 times in the last 4 hours (4 aggregated points)

Evaluated based on

Aggregation granularity (Period) *

1 hour

Frequency of evaluation

Every 5 Minutes

Advanced settings

Number of violations to trigger the alert: ⓘ

Number of violations

4

Evaluation period

4 hours

(4 aggregated points)

Ignore data before

Use this option to set the date from which to start learning the metric historical data and calculate the dynamic thresholds.

[Learn more about Dynamic Thresholds](#)

MM/DD/YYYY



h:mm A

Discovering Application Insights Smart Detection and Dynamic Thresholds

Home > Monitor >

Create alert rule

X

Send notifications or invoke actions when the alert rule triggers, by selecting or creating a new action group. [Learn more](#)

Action group name

casweetsag

Contains actions

1 Email ⓘ



[Manage action groups](#)

Alert rule details

Provide details on your alert rule so that you can identify and manage it later.

Alert rule name * ⓘ

rule1



Description

Specify the alert rule description



Save alert rule to resource group * ⓘ

casweetsrg



Severity * ⓘ

3 - Informational



Enable alert rule upon creation



Smart Detection rules for apps

Home > Monitor > ctlaltsweetsai >

Smart Detection settings



Smart Detection rules

ENAB...	NAME	INFO	SEVERITY
✓	Slow page load time	ⓘ	Information
✓	Slow server response time	ⓘ	Information
✓	Long dependency duration	ⓘ	Information
✓	Degradation in server response time	ⓘ	Information
✓	Degradation in dependency duration	ⓘ	Information
✓	Degradation in trace severity ratio (preview)	ⓘ	Information
✓	Abnormal rise in exception volume (preview)	ⓘ	Information
✓	Potential memory leak detected (preview)	ⓘ	Information
✓	Potential security issue detected (preview)	ⓘ	Information
✓	Abnormal rise in daily data volume (preview)	ⓘ	Information

Smart Detection rules for apps

Home > Monitor > ctlaltsweetsai > Alert rules >

Failure Anomalies - ctlaltsweetsai

Edit alert rule

Save Discard Disable Delete Properties

Scope

Select the target resource you wish to monitor.

Resource

ctlaltsweetsai

Condition

Configure when the alert rule should trigger by selecting a signal a

Condition name

Failure Anomalies detected

Add condition

Alert rules with a smart detector signal can include only one cond

Select a signal

X

Failure Anomalies

Detects if your application experiences an abnormal rise in the rate of HTTP requests or dependency calls that are reported as failed. The anomaly detection uses machine learning algorithms and occurs in near real time, therefore there's no need to define a frequency for this signal.

To help you triage and diagnose the problem, an analysis of the characteristics of the failures and related telemetry is provided with the detection. This feature works for any app, hosted in the cloud or on your own servers, that generates request or dependency telemetry - for example, if you have a worker role that calls [TrackRequest\(\)](#) or [TrackDependency\(\)](#).

[Learn more about Failure Anomalies](#)

A note about your data privacy:

The service is entirely automatic and only you can see these notifications. [Read more about data privacy](#)

Smart Alerts conditions can't be edited or added for now.

Discovering Application Insights Smart Detection and Dynamic Thresholds

- ➊ Dynamic Thresholds apply machine learning to determine the proper, appropriate metrics to be used as thresholds.
- ➋ Smart Detection applies machine learning toward application telemetry to notify you of anomalies.

Deciding Which Dependencies to Set Alerts On

Deciding Which Dependencies to Set Alerts On

[Application Insights Dependency Tracking](#)

[Which Dependencies Are Tracked in Application Insights?](#)

[Where Can I Find Dependency Data?](#)

[Application Dependencies on Virtual Machines](#)

[Demo: Exploring Dependencies](#)

Deciding Which Dependencies to Set Alerts On

What Is a Dependency?

One component relies on another component to perform some function.

Dependencies exist because each component brings something unique.



HTTP, database, and file system calls



Internal vs. External



Strong vs. Weak

Deciding Which Dependencies to Set Alerts On

Application Insights Dependency Tracking

1 Track and Monitor

Helps identify strong dependencies by tracking and monitoring calls

3 Manual Dependency Tracking

Configured using the `TrackDependency` API

2 Automatic Tracking with .NET/.Net Core

Tracking is configured by default if using the .NET and .NET Core SDK for Application Insights

4 AJAX from Web Pages

Application Insights JavaScript SDK will collect AJAX calls automatically

Which Dependencies are tracked in Application Insights

Automatic

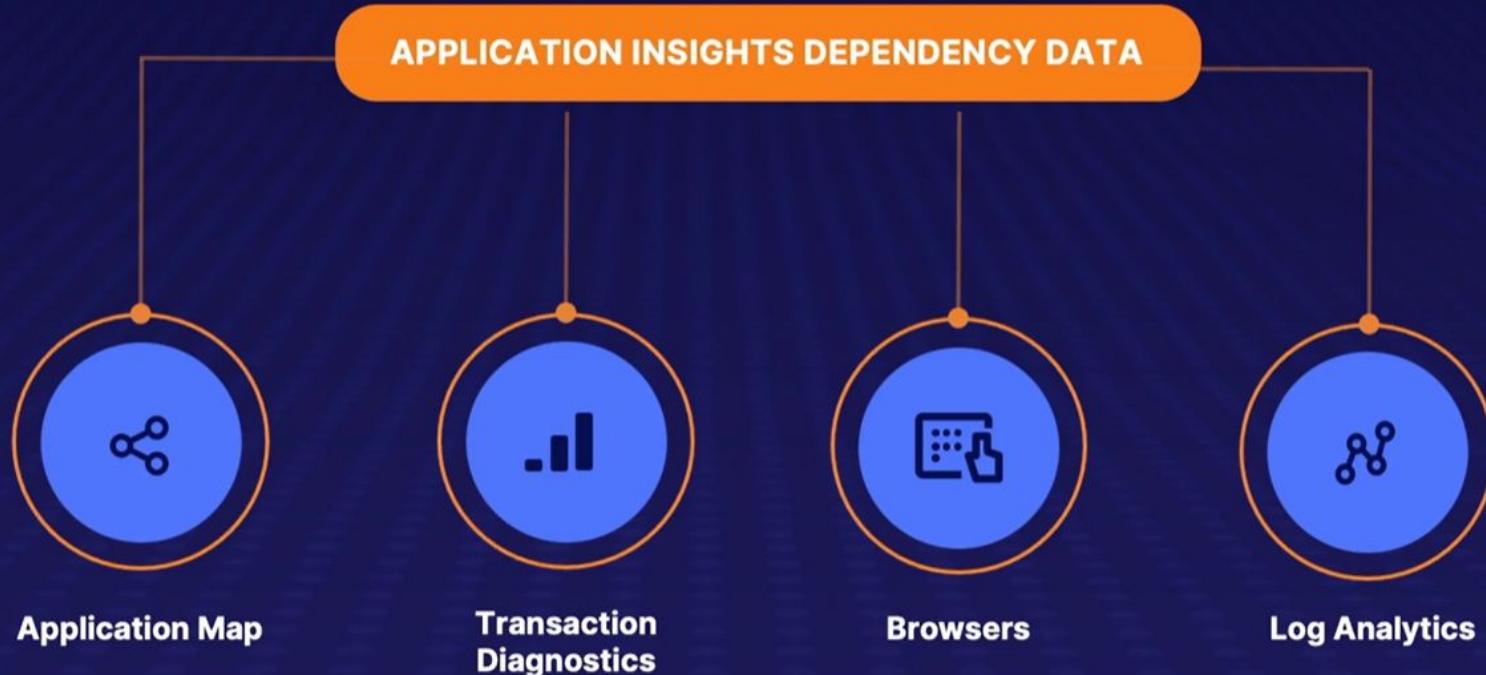
- HTTP and HTTPS calls
- WCF if using the HTTP bindings
- SQL calls using SqlCommand
- Azure Storage with Azure Storage Client
- EventHub Client SDK
- ServiceBus Client SDK
- Azure Cosmos DB with HTTP/HTTPS

Manual

- Cosmos DB with TCP
- Redis

VS

Where can i find Dependency data?



Deciding Which Dependencies to Set Alerts On



Agent

Dependency agent needs to be installed.



Views

Different scopes of views can be seen from individual virtual machines (VMs), VM scale sets, or from Azure Monitor.



Processes

Shows dependencies by looking for processes that are running with:

- Connections between the servers that are active
- Any inbound/outbound connection latency
- TCP-connected ports

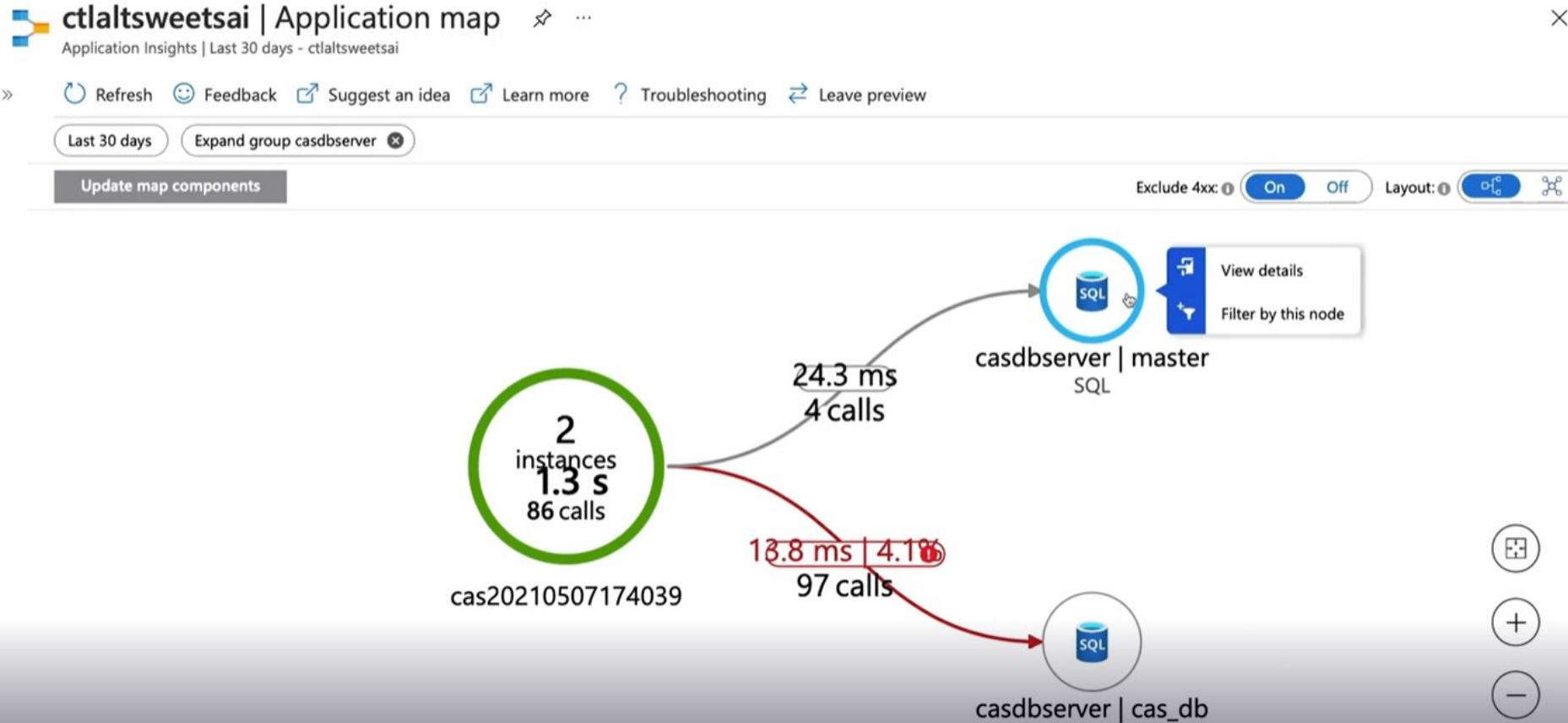


Connection Metrics

- Response time
- How many requests

Deciding Which Dependencies to Set Alerts On

Home > Monitor > ctlaltsweetsai



Deciding Which Dependencies to Set Alerts On

ctlaltsweetsai | Application map ⚡ ...

Application Insights | Last 30 days - ctlaltsweetsai

Refresh Feedback Suggest an idea Learn more Troubleshooting Leave preview

Last 30 days Expand group casdbserver

Update map components

Exclude 4xx: On Off Layout:

cas20210507174039

2 instances 1.3 s 86 calls

13.8 ms | 4.1% 97 calls

cas202105071... → casdbserver | SQL

View in Logs View casdbserver

Collapse all

TOP FAILING STATUS CODES

STATUS CODE	COUNT
208	4

Investigate failures

SLOWEST CALLS BY NAME

NAME	DURATION (A...)
SQL: tcp:casdbserver.database.windows....	13.8 ms

The screenshot shows the Microsoft Application Insights Application Map interface. It displays a network of dependencies between application components. A central component, 'cas20210507174039', is highlighted with a green circle and shows performance metrics: 2 instances, 1.3 seconds total time, and 86 calls. A dependency arrow points from this component to 'casdbserver' (SQL), which is also highlighted with a green circle. This dependency shows 97 calls with a duration of 13.8 ms and a failure rate of 4.1%. To the right of the map, there are detailed sections for 'TOP FAILING STATUS CODES' (showing 208 with a count of 4) and 'SLOWEST CALLS BY NAME' (showing a single slow call to 'SQL: tcp:casdbserver.database.windows....' with a duration of 13.8 ms). Navigation and search filters are visible at the top left.

Deciding Which Dependencies to Set Alerts On

Home > Monitor > ctlaltsweetsai >

Failures

X



Refresh

View in Logs

Analyze with Workbooks

Copy link

Feedback

Server

Browser

target has "tcp:casdbserver.database.windows.net,1433 | cas_db"



Operations Dependencies Exceptions Roles

Failed dependency count



Select operation

Search to filter items...

DEPENDENCY NAME

COUNT (FAILED) ↑ COUNT ↓ PIN

Overall

Top 10 response codes

COUNT

FILTER...

208

4

» Select a sample dependency

Filtered on client_Type !...

target has == "tcp:casdbse..."

Success == false

Suggested

5/7/2021, 1:50:01 PM
tcp:casdbserver.database.windows.net,
1433
Duration: 45 ms Response code: 208
Dependency name:
SQL: tcp:casdbserver.database.windows.net,1433
| cas_db

All

5/7/2021, 1:50:01 PM
tcp:casdbserver.database.windows.net,
1433
Duration: 45 ms Response code: 208
Dependency name:
SQL: tcp:casdbserver.database.windows.net,1433
| cas_db

Sort by

Relevance

Deciding Which Dependencies to Set Alerts On

Home > Monitor > ctlaltsweetsai > Failures >

End-to-end transaction details

Search results

Filtered on

timestamp > 4/12/2021, 2:...

timestamp < 5/12/2021, 2:...

client_Type !...

target has == "tcp:cldbser..."

Success == false

Suggested

5/7/2021, 1:50:01 PM

tcp:cldbserver.database.windows.net,1433

Duration:45 ms Response code:208 →

Dependency name:

SQL: tcp:cldbserver.database.windows.net,1433 | cas_db

Sort by ⓘ

Relevance ▾

All

5/7/2021, 1:50:01 PM

tcp:cldbserver.database.windows.net,1433

Duration:45 ms Response code:208 →

« Search results Learn more Copy link Feedback ▾ Leave preview »

End-to-end transaction
Operation ID: 4cdf2c2a378269468873f5335d9fb078

Dependency (outgoing) Request (incoming)

EVENT	TIME	DETAILS
cas20210507174039	GET Todos/Index	casdbserver cas_db
		casdbserver master
		casdbserver cas_db
		casdbserver master
		casdbserver cas_db

Create work item ▾

casdbserver
cas_db

Dependency type	SQL
Result code	208
Dependency call status	false
Dependency duration	45 ms
Remote dependency name	SQL: tcp:cldbserver.database.windows.net,1433 cas_db

Command Copy
tcp:cldbserver.database.windows.net,1433 | cas_db

Related Items

Show what happened before and after this dependency in User Flows

Deciding Which Dependencies to Set Alerts On

Home > Monitor > ctlaltsweetsai >



Performance



12 ms

10ms

Dependency count

50

0

Sun 18

Sun 25

Sun 2

Sun 9

02:38 PM

02:38 PM

Select operation

Search to filter items...

DEPENDENCY NAME

DURATION (AVG) ↑ COUNT ↑ ⏪ PIN

Overall

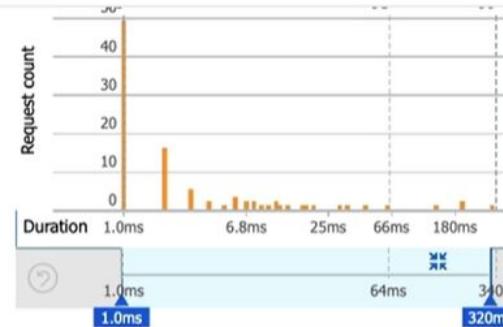
13.8 ms

97

SQL: tcp:cassandra.database.windows.net...

13.8 ms

97



Insights (2)

85% COMMON PROPERTIES:
client_City, client_CountryOrR...

99% COMMON PROPERTIES:
client_CountryOrRegion, perfor...

» Select a sample dependency

Filtered on

Dependency name == SQL: tcp:cas...

client_Type !...

target has == "tcp:cassandra...

Suggested

5/10/2021, 9:25:07 AM

tcp:cassandra.database.windows.net,
1433

Duration: 1 ms Response code:

Dependency name:

SQL: tcp:cassandra.database.windows.net,1433
| cas_db

All

5/10/2021, 9:25:07 AM

tcp:cassandra.database.windows.net,
1433

Duration: 1 ms Response code:

Dependency name:

Sort by ⓘ

Relevance ▾

Deciding Which Dependencies to Set Alerts On

Home > Monitor > ctlaltsweetsai >

Logs X

ctlaltsweetsai

New Query 1* + New query

Feedback Queries Query explorer

ctlaltsweetsai Select scope Cancel Time range : Set in query Save Share New alert rule Export Pin to dashboard ...

Tables Queries Functions

Search ...

Filter Group by: Solution ...

Collapse all

Favorites

You can add favorites by clicking on the ★ icon

Application Insights

- ▶ grid availabilityResults
- ▶ grid browserTimings
- ▶ grid customEvents
- ▶ grid customMetrics

```
1 // Count of failed and successful calls over time
2 let safeDependency = datatable(
3     id: string,
4     target: string,
5     type: string,
6     name: string,
7     data: string,
8     success: string,
9     resultCode: string,
10    duration: double,
11    performanceBucket: string,
```

Results Chart ...

We're getting your data, hang in there... 🕒 00:00 0 records

Deciding Which Dependencies to Set Alerts On

Home > Monitor > ctlaltsweetsai >

Logs ...

ctlaltsweetsai

Select scope Run Time range : Set in query Save Share New alert rule Export Pin to dashboard ...

Tables Queries Functions ...

Search Filter Group by: Solution

Collapse all

Favorites

You can add favorites by clicking on the star icon

Application Insights

- availabilityResults
- browserTimings
- customEvents
- customMetrics
- dependencies
- exceptions
- pageViews

```
1 // Count of failed and successful calls over time
2 let safeDependency = datatable(
3     id: string,
4     target: string,
5     type: string,
6     name: string,
7     data: string,
8     success: string,
9     resultCode: string,
10    duration: double,
11    performanceBucket: string,
```

Results Chart Display time (UTC+00:00) 00:01.0 3 records

Completed

timestamp [UTC]: 5/7/2021, 12:00:00.000 AM succeeded: 19

0 50
Calls

May 8 May 10 May 12

timestamp [UTC]

Chart format

Deciding Which Dependencies to Set Alerts On

Home > Monitor

Monitor | Virtual Machines

Microsoft

Search (Cmd +/)

Refresh

Provide Feedback

Get Started

Performance

Map

Subscription: ACG TA - Alex Potasnick

Resource Group: All

Type: All

Workspace: la-2juwmhr22al4i

View Workbooks

1.4MB

976.6KB

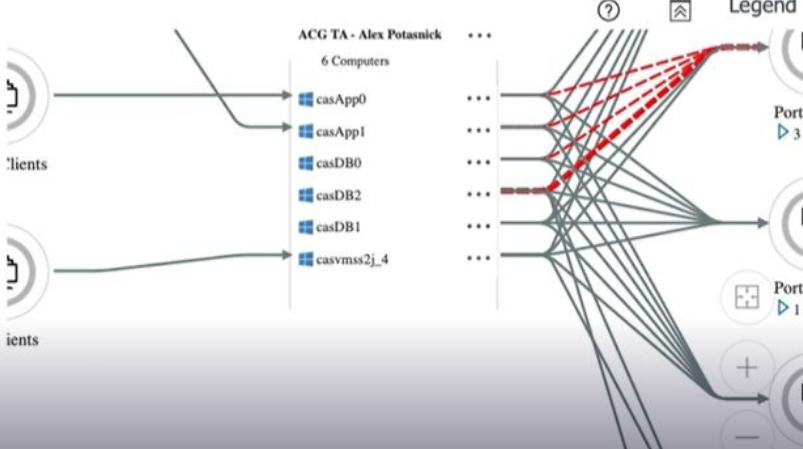
488.3KB

0B

Sent
7.8 KB

Received
75.2 KB

Time range: Last 30 minutes as of 12 May 14:49



Links 1 minute granularity

12

10

8

6

4

2

0

Failed
32.26 m

Live
3.39

Established
6.68

Terminated
6.68

1 minute granularity

Deciding Which Dependencies to Set Alerts On

- ✓ Dependencies are components in an application that rely on each other for specific functions.
- ✓ Dependency Tracking is automatically configured with the .NET and .NET Core SDK for Application Insights.
- ✓ Manual dependency tracking can be configured using the `TrackDependency` API.

Deciding Which Dependencies to Set Alerts On

- ✓ The Application Map provides visualizations of application dependencies and correlated data.
- ✓ A virtual machine application dependency map can be found in Azure Monitor with system information and connection metrics.

Summary

Big Picture: What Is Site Reliability Engineering (SRE)?

Key Learning Outcomes

Reliability

- Application is up when you expect it to be
- SLA
- SLO
- SLI

Key Concepts

- Feedback
- Measure everything
- Alerting
- Automation
- Small changes
- Embrace risk

SRE vs. DevOps

- SRE focuses on production
- DevOps focuses on deployments

Summary

Exploring Metric Charts and Dashboards

Key Learning Outcomes

Azure Monitor

- Metrics
- Logs

Monitor Metrics

- Visualize
- Analyze
- Alert
- Automate

Charts

- Scope
- Namespace
- Metric
- Aggregation
- Time period

Summary

Implementing Application Health Checks

Key Learning Outcomes

Application Insights

- How the application is doing
- How it is being used

Availability

- URL ping test

Alerting

- Notify on failures
- Action Groups

Summary

Exploring System Load and Failure Conditions

Key Learning Outcomes

Failure Mode Analysis

- Failure points and modes
- Risk and severity
- Response

Failure Reduction

- Understand the application and the environment
- Implement redundancy

Performance Testing

- Load Testing
- Stress Testing

Summary

Understanding Baseline Metrics

Key Learning Outcomes

Baseline Tools

- Log Analytics
- Metrics Explorer

Insights

- Azure Monitor Insights
- Application Insights

Summary

Discovering Application Insights Smart Detection and Dynamic Thresholds

Key Learning Outcomes

Dynamic Thresholds

- Machine learning for metrics

Smart Detection

- Machine learning for application telemetry

Categories

- Failure alerts
- Performance notifications

Summary

Deciding Which Dependencies to Set Alerts On

Key Learning Outcomes

Dependencies

- Application components that rely on each other for specific functions

Tracking

- Automatic with .NET and .NET Core SDK for Application Insights
- Manually configured with TrackDependency API

Map

- Application Insights Application Map
- Virtual Machine Map

Designing and Implementing a Source Control Strategy

Introduction to Source Control

What Is Source Control?

- ✓ Also known as source repositories/version control
- Central **source of truth** for group codebase
- ✓ Allows multiple developers to collaborate on code and track changes
- Critical for any multi-developer project

Examples:



Azure Repos

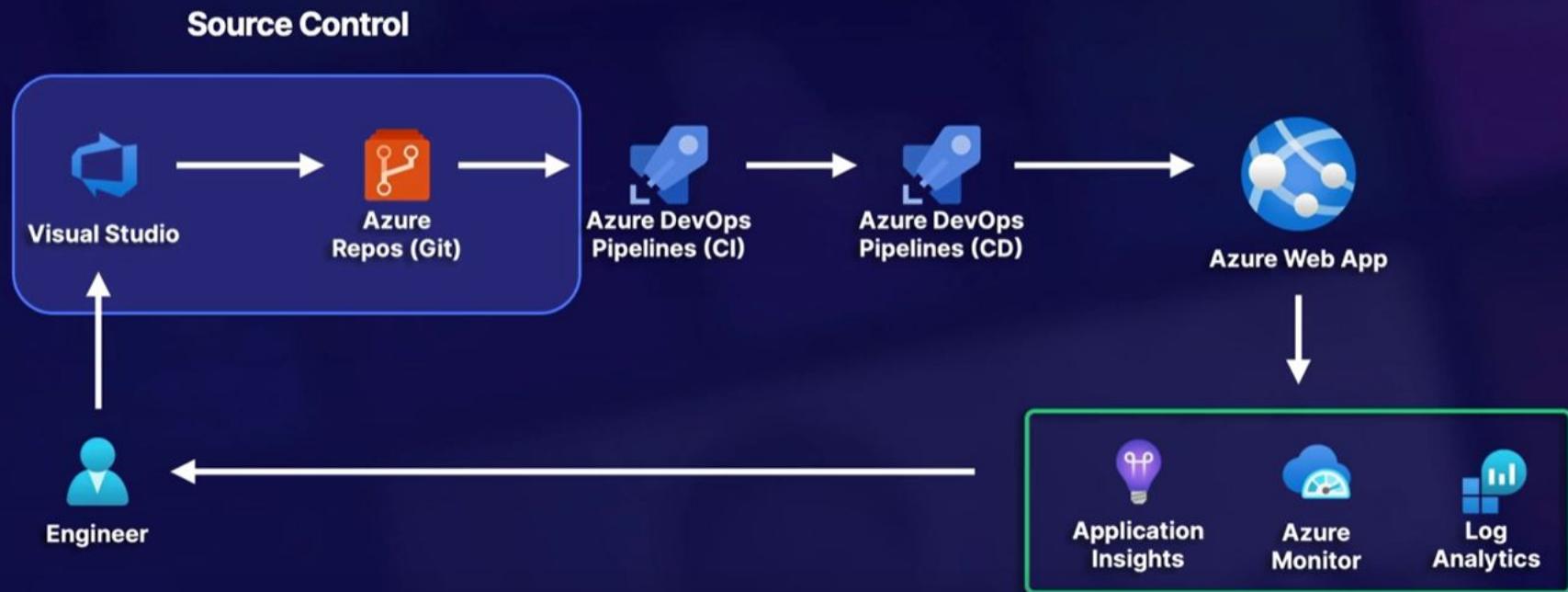


GitHub



Bitbucket

Introduction to Source Control



Introduction to Source Control

Source Control Types



Git

Distributed/decentralized



Team Foundation Version Control (TFVC)

Centralized/client-server

Introduction to Source Control

Git: The Default/ Preferred Option

- ✓ Each developer has a **copy** of the repo on their local machine.
Includes all branch and history information.
- ✓ Each developer checks in their local portion of the code, and changes are **merged** in a central repository.

Introduction to Source Control

TFVC: The Non-Default Option



Developers **check out** the only version of each file on local machines.



Checked-in code is then pushed to everyone else.

Introduction to Source Control

Which One Should I Use?



Per Microsoft: *Git is preferred unless there is a specialized need for centralized version control in TFVC.*

As we move along, we will cover:

- Working with repos inside of Azure Repos
- General Git concepts
- Connecting repos to Pipelines (later)

Exploring Azure Repos

Azure Repos at a Glance

Managed Repositories



Exist inside of Azure DevOps Organization



Project level



Supports Git and TFVC



Optional component for Azure Pipelines

Can use external repos in Pipelines

Setting Up Azure Repos

Multiple Options



All options involve getting code from *somewhere else* into Azure Repos

Azure Repos then becomes the **source of truth**



```
from flask import Flask  
app = Flask(__name__)  
  
@app.route("/")  
def hello():  
    return "Hello, World!"
```



Azure Repos

Exploring Azure Repos

Import Options



**Set up empty repo
from scratch**



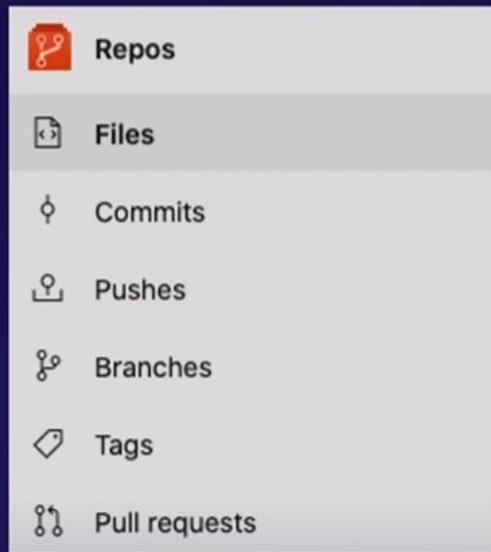
**Clone existing repo into Azure Repo
(GitHub, another Azure Repo)**



**Push local codebase
into Azure Repo**

Exploring Azure Repos

Azure Repos Supports All Git Features



- Branching Methods
- History
- Tagging
- Pull Requests
- And much more!

If it works in Git, it works in Azure Repos!

Repository Sharing with Submodules

Why do we care about this?

1 Challenge: Need to incorporate resources from a different Git project in your current repo

- ✓ Examples: Third-party library used in your projects
- ✓ Need to treat external resources as separate entities, yet seamlessly included within your project

2 Solution: Submodules

- ✓ Not limited to Azure Repos, core Git feature — with an Azure twist!



Repository Sharing with Submodules

How it Works

1 Add the external repo as a submodule to your repo.

```
git submodule add  
https://github.com/account/added-package
```

2

When cloning a project with a submodule into your local directory, extra steps are required:



Initialize and update submodule.

```
git submodule init  
git submodule update
```

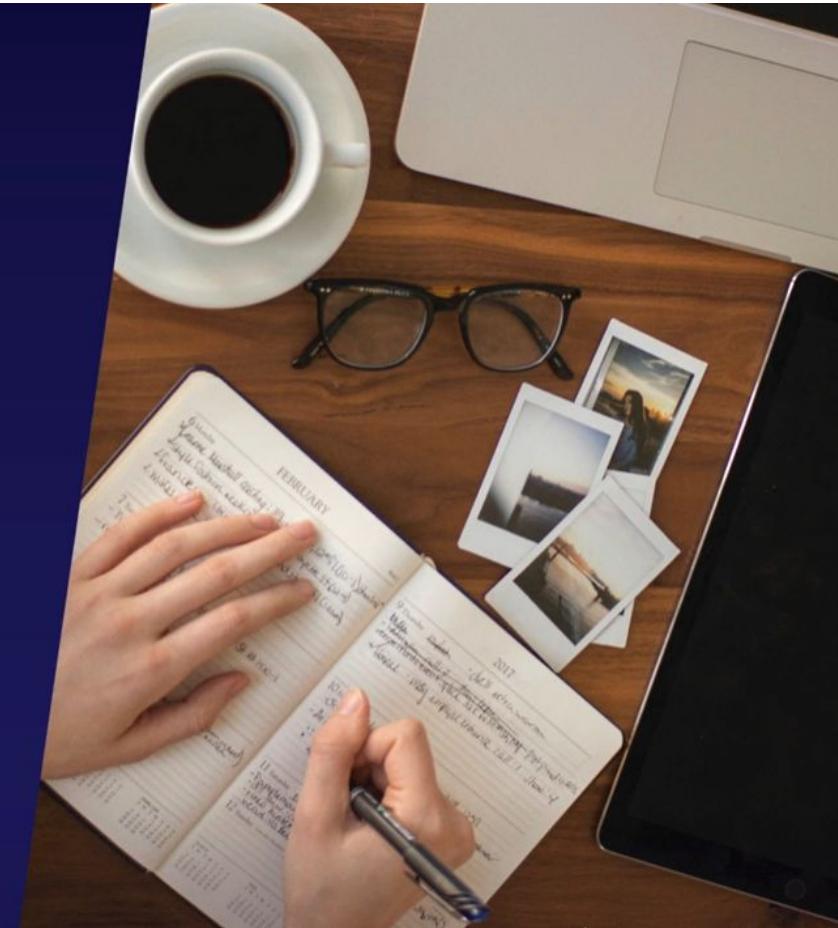
Repository Sharing with Submodules

Submodules in Azure Pipelines

1

Requirements to include in build pipelines:

- ✓ **Unauthenticated** - i.e., publicly available
- ✓ **Authenticated within your main repository's account**
 - Same GitHub organization, Azure organization, etc.
 - Same account to access the primary repo must also be the same one to access the submodule repository
 - Submodules must be registered via HTTP — not SSH



Scalar



What Is Scalar?

Git client tool used to optimize new and existing repositories

Git was not designed to grow to certain sizes, so you might notice Git becoming sluggish when projects get too large, which makes it difficult to scale.

Scalar



Scalar

```
#Clone a new repository  
scalar clone [options] <url> [<dir>]
```

```
#Register an existing repository  
scalar register [<path>]
```

OPTION 1

NEW REPOSITORY

- Create repository
- Use partial clone feature
- Use sparse checkout feature
- Set up Git maintenance
- Configure optimization settings

OPTION 2

EXISTING REPOSITORY

Apply features and settings to optimize existing repository

Scalar

- Scalar is a Git tool that helps improve performance on large repositories.
- `scalar clone` is used for new repositories.
- `scalar register` is used for existing repositories.

Connecting to GitHub Using Azure Active Directory

GitHub and Azure Active Directory (AAD) Integration

1

Why This Matters

- ✓ By default, GitHub and AAD identities are separately maintained.
- ✓ However, we can integrate GitHub identities with AAD using **Single Sign On (SSO)**.

2

Advantage of GitHub/AAD Integration

- ✓ Manage GitHub accounts from a central location:
Azure Active Directory.



GitHub



Azure Active
Directory

Requirements for GitHub/AAD SSO

1

GitHub Enterprise Cloud Organization



GitHub Team plan unable to use SSO

2

Permissions



GitHub: Administrator



Azure: Create SSO – Global Admin, Cloud Application Administrator, Application Administrator



How it works: Azure AD SSO Configuration

1

Add GitHub in Enterprise Applications.

2

Configure SAML SSO configuration with GitHub Enterprise account.

Link to GitHub Enterprise Organization.

Set User Attributes.

Download Base64 Signing Certificate (for GitHub side).

Generate GitHub links to AAD.

3

Add AAD Users to GitHub SSO.

The screenshot shows the Azure AD SSO configuration interface with four numbered steps:

- 1 Basic SAML Configuration**

Identifier (Entity ID)	Required
Reply URL (Assertion Consumer Service URL)	Required
Sign on URL	Optional
Relay State	Optional
Logout URL	Optional
- 2 User Attributes & Claims**

givenname	user.givenname
surname	user.surname
emailaddress	user.mail
name	user.userprincipalname
Unique User Identifier	user.userprincipalname
- 3 SAML Signing Certificate**

Status	Active
Thumbprint	9CEA37643ACE0D710AD63296857B251D1FC5C48
Expiration	12/20/2025, 2:50:17 PM
Notification Email	matthew@ctrl-alt-sweets.com
App Federation Metadata Url	https://login.microsoftonline.com/14ffe2d-e4f6-4...
Certificate (Base64)	Download
Certificate (Raw)	Download
Federation Metadata XML	Download
- 4 Set up GitHub Enterprise Cloud - Enterprise Account**

You'll need to configure the application to link with Azure AD.

Login URL	https://login.microsoftonline.com/14ffe2d-e4f6-4...
Azure AD Identifier	https://sts.windows.net/14ffe2d-e4f6-49fe-95a4-...
Logout URL	https://login.microsoftonline.com/14ffe2d-e4f6-4...

How it works: Github Enterprise Configuration

1 Enable SAML Authentication.

2 Configure link to AAD tenant.

Login URL → Sign on URL

AAD Identifier → Issuer

Open and copy/paste Signing
Certificate from AAD

Set Signature/Digest Method to
RSA-SHA256/SHA256

Sign on URL
Members will be forwarded here when signing in to your organization

Issuer
Typically a unique URL generated by your SAML Identity Provider

Public certificate

Your SAML provider is using the RSA-SHA1 Signature Method and the SHA1 Digest Method

The assertion consumer service URL is <https://github.com/orgs/> [REDACTED] /saml/consume.

Incorporating Changelogs

What Are Git Changelogs?

1 Record of changes (commits) in a project's lifetime

2 Why do we care about changelogs?

- ✓ Keep running list of changes/updates
- ✓ Useful for teams working on a single project



Incorporating Changelogs

Manually Creating/Viewing Changelogs

1

Git log command

```
git log
```

2

Options to clean log output



One-line summaries



Remove commit ID, custom elements

```
git log --oneline
```

```
git log --pretty="- %s"
```

Incorporating Changelogs

Automation Options

Third-Party Applications

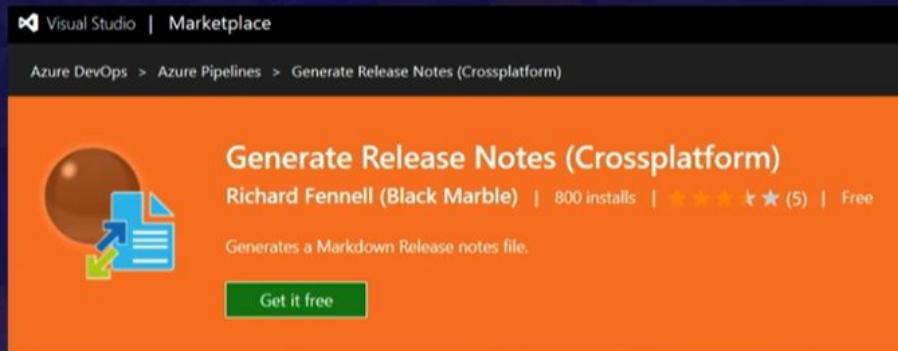
GitHub Changelog Generator
Auto Changelog

IDE Plugins

Example: Visual Studio changelog plugin

Pipeline Plugins

Example: Jenkins has a changelog plugin



Summary

Introduction to Source Control

Key Learning Outcomes

Source Control Types

- Git
- TFVC

Differences

- Git = decentralized
- TFVC = centralized

Which Should You Use?

- Default to Git unless there is a need for centralization

Summary

Exploring Azure Repos and Demo

Key Learning Outcomes

What Is Azure Repos?

- Managed repos inside Azure DevOps organization (project level)

Code Import

- Clean setup
- Clone from external source
- Push local code

Git Workflow

- Authenticate to Azure Repos
- Adds, commits, push

Summary

Repository Sharing with Submodules

Key Learning Outcomes

Purpose

- Incorporate external (and separately maintained) resources in your repo

Azure Considerations

- Unauthenticated source
- Authenticated source with same permissions as primary repo

Summary

Discovering Scalar

Key Learning Outcomes

Scalar

- Optimize and manage large repos

Advantages

- Data transfer
- Command runtime
- Relevant files
- Organization

Commands

- `scalar clone`
- `scalar register`

Summary

Connecting to GitHub Using Azure Active Directory

Key Learning Outcomes

Purpose	Requirement	Process
<ul style="list-style-type: none">• Manage SSO to GitHub via Azure Active Directory (Azure AD)	<ul style="list-style-type: none">• GitHub Enterprise Cloud	<ul style="list-style-type: none">• Configure SAML SSO in both Azure AD and GitHub

Summary

Incorporating Changelogs

Key Learning Outcomes

What Are Changelogs?

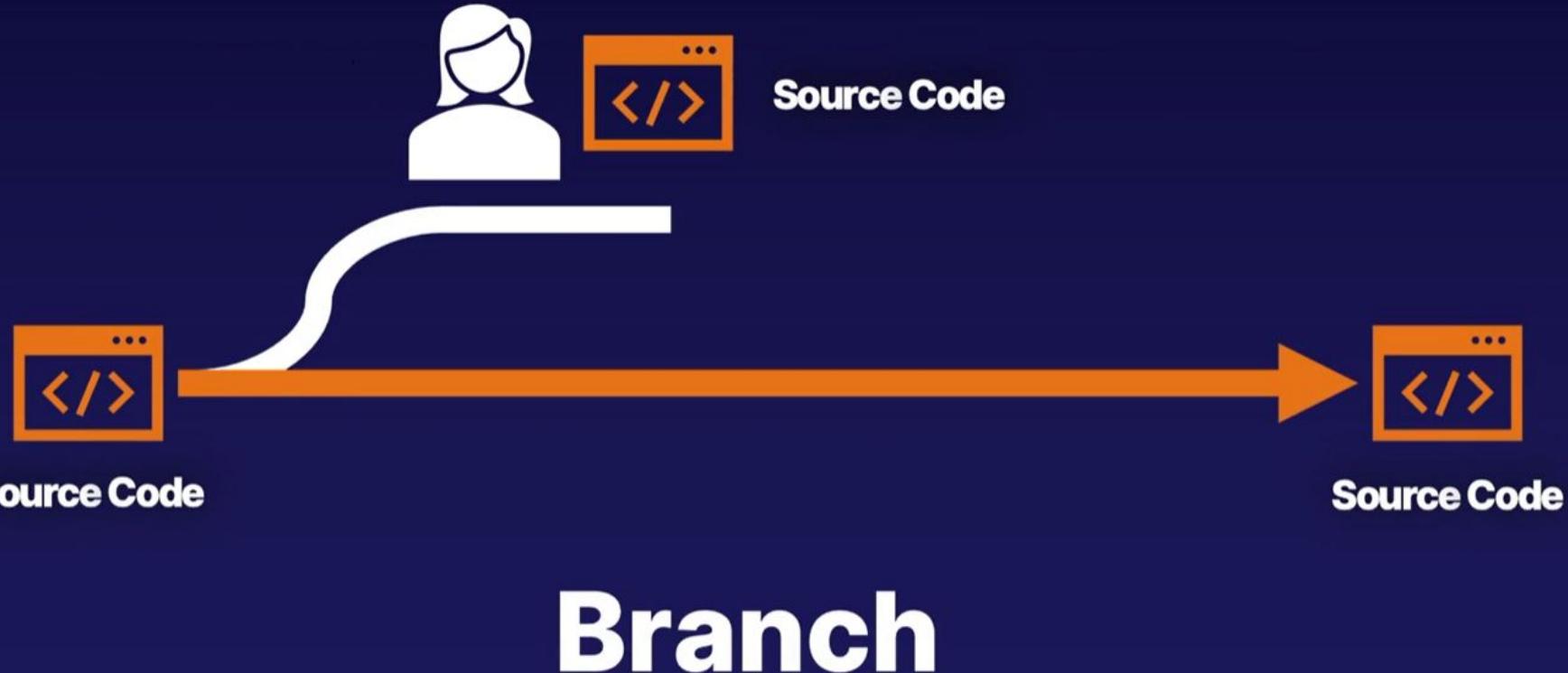
- Record of changes (commits) in project lifetime
- View via Git logs

Changelog Automation

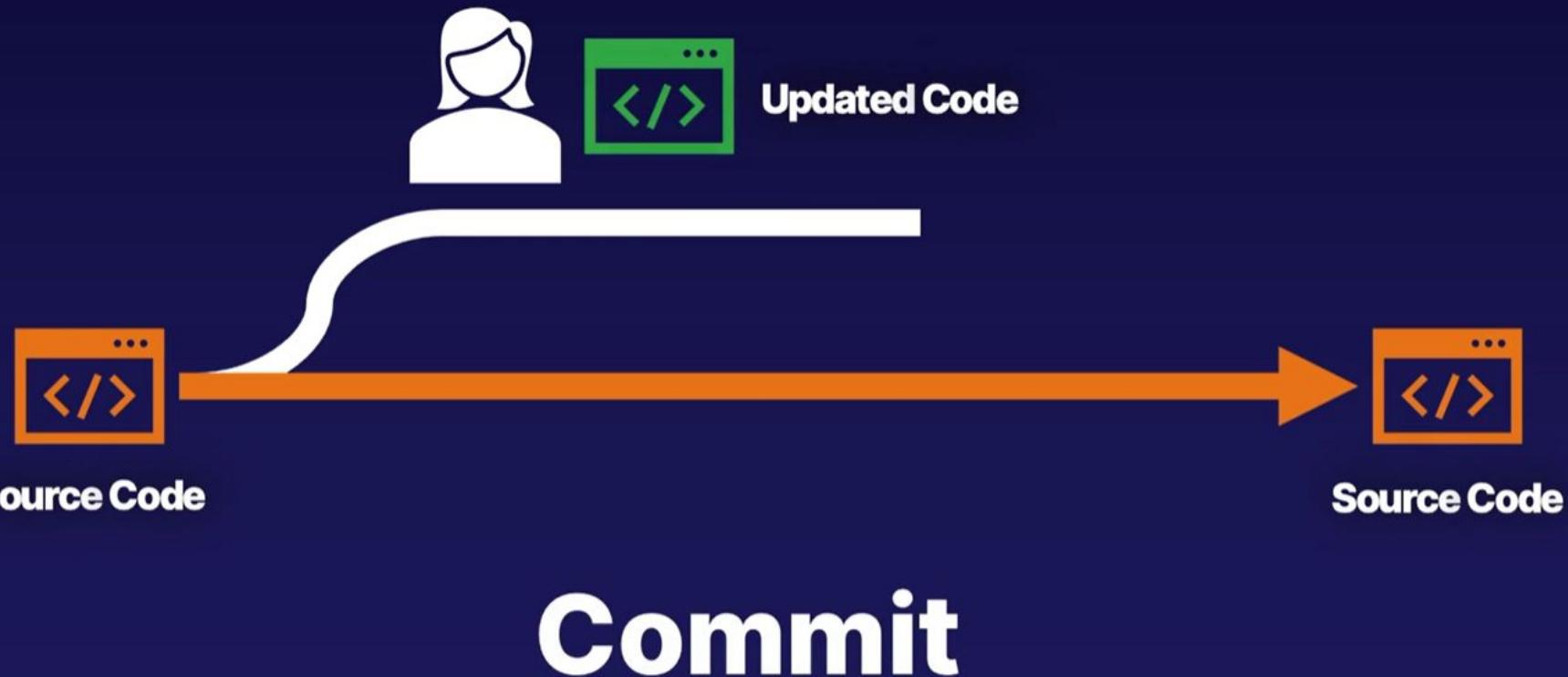
- Third-party applications
- IDE plugins (Visual Studio)
- Pipeline plugins (Jenkins)

Planning and Implementing Branching Strategies for the Source Code

Configuring Branches



Configuring Branches



Configuring Branches

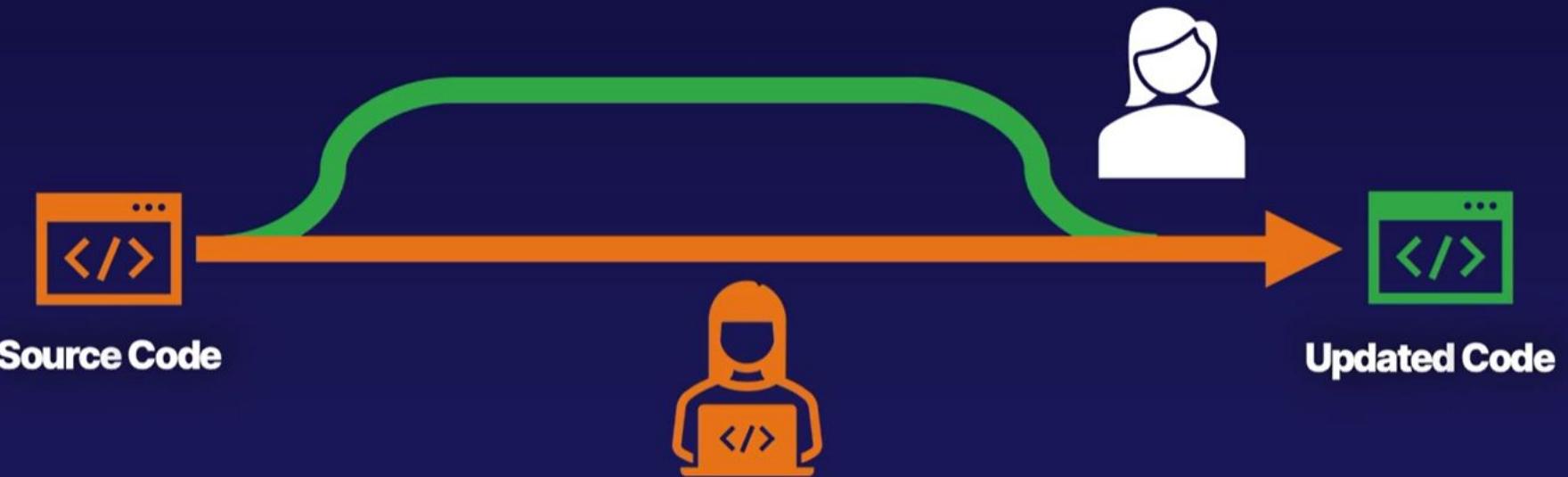


Pull Request

Configuring Branches



Configuring Branches



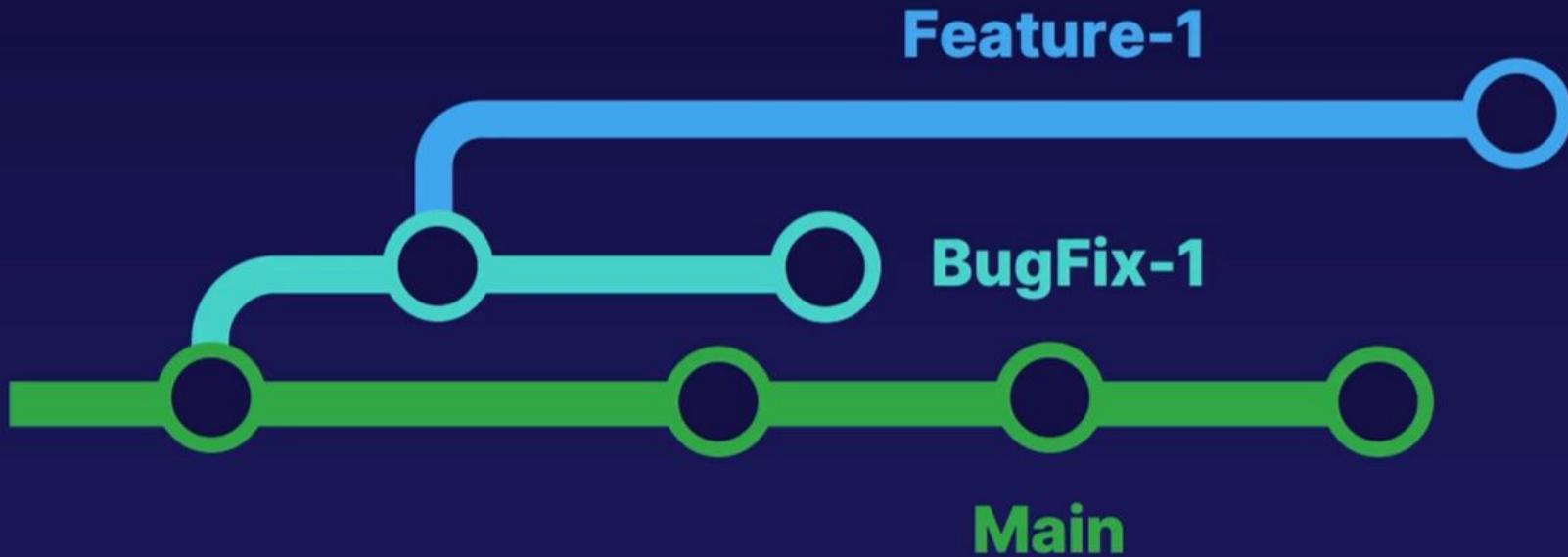
Push, Fetch, and Pull

Configuring Branches



Isolate a set of changes from the rest of the code

Configuring Branches



Configuring Branches

Branch Management



Branch Policies

Initial safeguards



Branch Restrictions

Advanced safeguards



Branch Protections

Minimize catastrophic actions

Configuring Branches

1

Require a Minimum Number of Reviewers

Require approval from a specified number of reviewers on pull requests.

3

Check for Comment Resolution

Check to see that all comments have been resolved on pull requests.

2

Check for Linked Work Items

Encourage traceability by checking for linked work items on pull requests.

4

Limit Merge Types

Control branch history by limiting the available types of merge when pull requests are completed.

Configuring Branches

5 Build Validation

Validate code by pre-merging and building pull request changes.

6 Status Checks

Require other services to post successful statuses to complete pull requests.

7

Automatically Included Reviewers

Designate code reviewers to automatically include when pull requests change certain areas of code (manual approvals).

8

Restrict Who Can Push to the Branch

Use security permissions to allow only certain collaborators the ability to push to the branch.

Configuring Branches

A

Prevent deletion
accidentally or
intentionally

B

Prevent overwriting
the branch commit
history with a force push

&

Discovering Branch Strategies

Why You Need a Branching Strategy



Optimize

Productivity



Enable

Parallel Development



Plan

Sets of structured releases



Pave

Promotion paths for software changes through production



Tackle

Delivered changes quickly



Support

Multiple versions of software and patches

Discovering Branch Strategies

Branching Strategies

Efficiently Manage Code Changes



Trunk-Based

Really, really quick branches



Feature (Task)

Branch per story

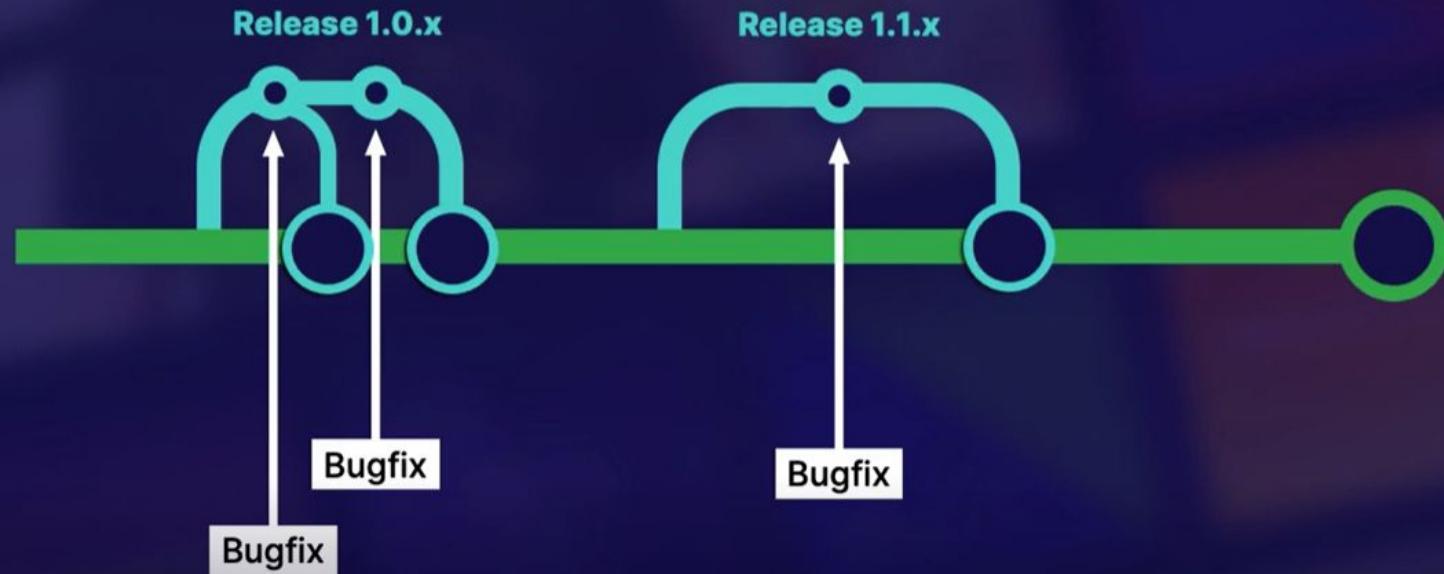


Release

All applicable user stories

Discovering Branch Strategies

Developers push directly to the main code



Discovering Branch Strategies

Pros

- Easier for a **REALLY** small number of developers

Cons

- Large code review process
 - Merge conflicts

VS

Discovering Branch Strategies

Branches are created for each feature or task



Discovering Branch Strategies



Discovering Branch Strategies

Use **flags** during development to make independent feature lifecycles



Discovering Branch Strategies

Branches are created for **all features** per release



Discovering Branch Strategies

Pros

- Supports multiple versions in parallel
- Customizations for a specific customer
- Focus on specific issues per patch or release

Cons

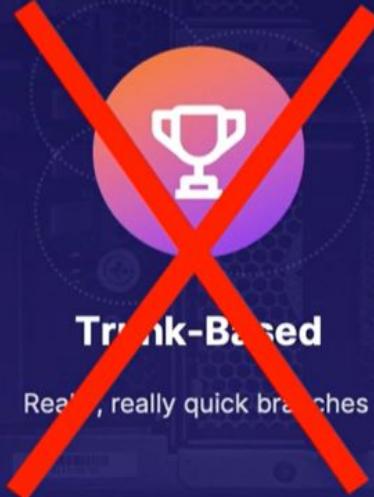
- Difficult to maintain
- Cannot have many changes or contributors
- Potentially create more work for teams per version

VS

Discovering Branch Strategies

Branching Strategies

Efficiently Manage Code Changes



Branch Strategies Summary



Trunk-based branching is not recommended.



Pick release branching if multiple or custom versions must be supported.

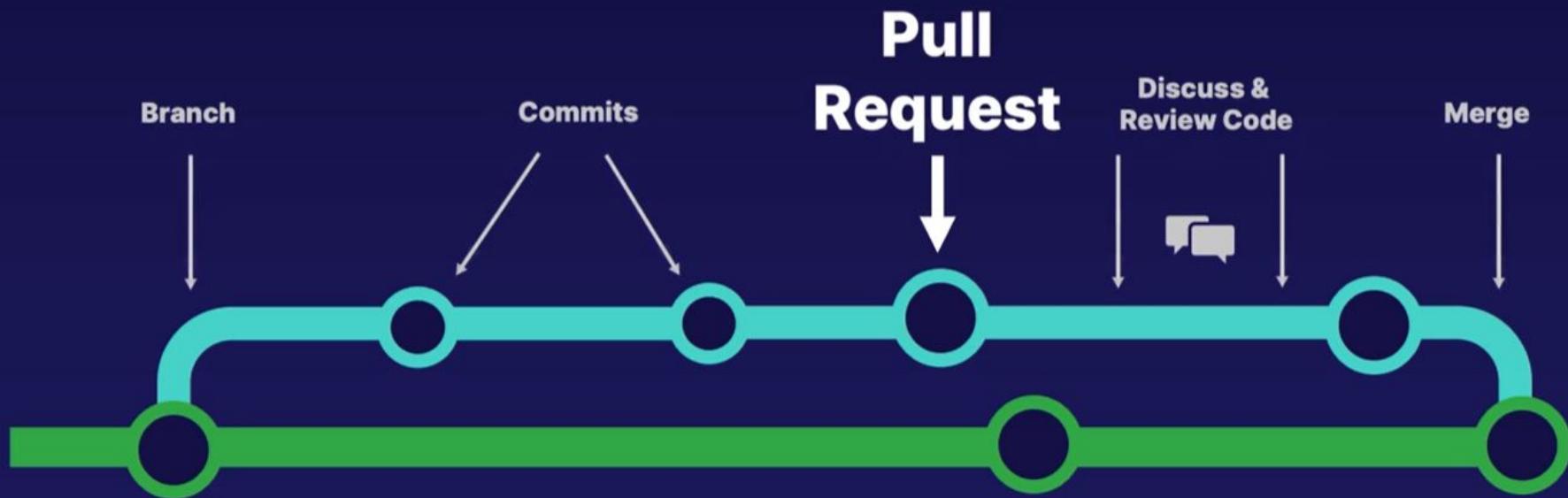


Feature branching enables CI/CD workflows.

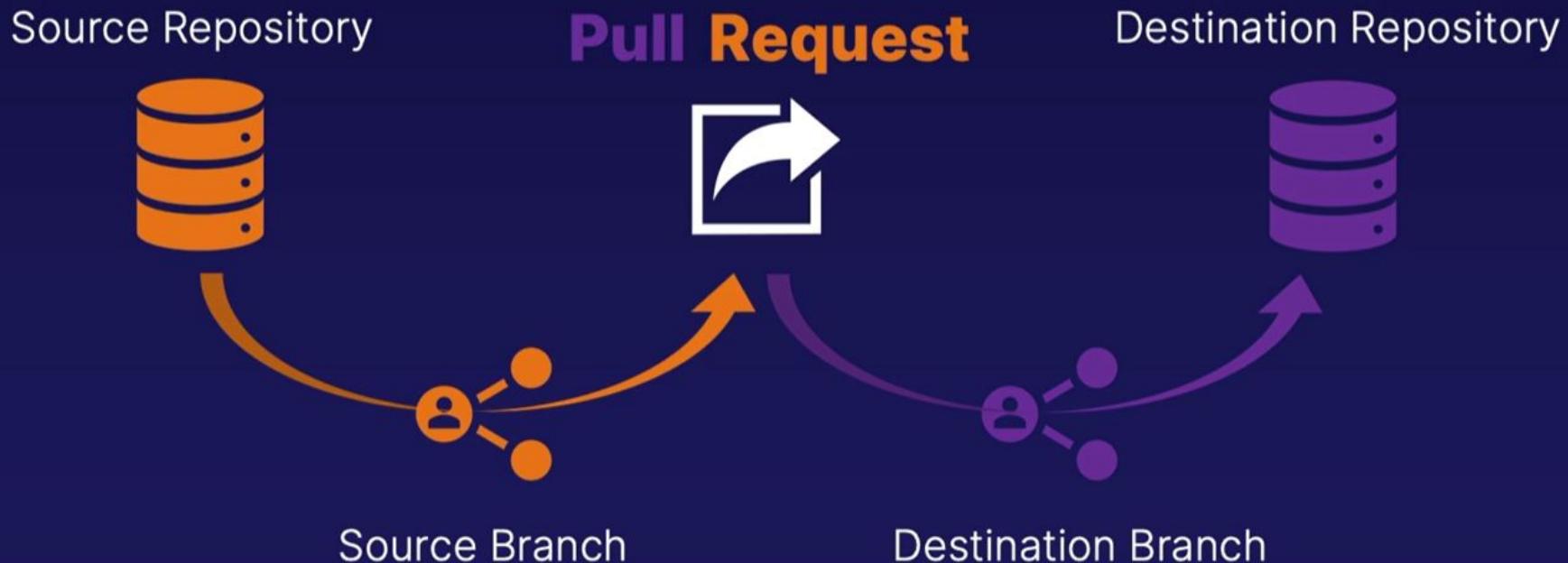


Flags (toggles) streamline feature lifecycle development.

Understanding the Pull Request Workflow



Understanding the Pull Request Workflow



Understanding the Pull Request Workflow

Understanding the Pull Request Workflow

Goals of a Pull Request

REDUCE BUG INTRODUCTION

Documentation and full transparency enable teams to verify changes before merging.

ENCOURAGE COMMUNICATION

Feedback and voting in a collaborative atmosphere, even in early development.

SPEED PRODUCT DEVELOPMENT

Faster and more concise process ensures speedy and accurate reviews.

Understanding the Pull Request Workflow

1 What

An explanation of the changes that you made
(context)

2 Why

The business or technical goal of the changes
(the bigger picture)

3 How

Design decisions and rationale on code
change approaches
(reasoning)

4 Tests

Verification of tests performed and all results
(verification)

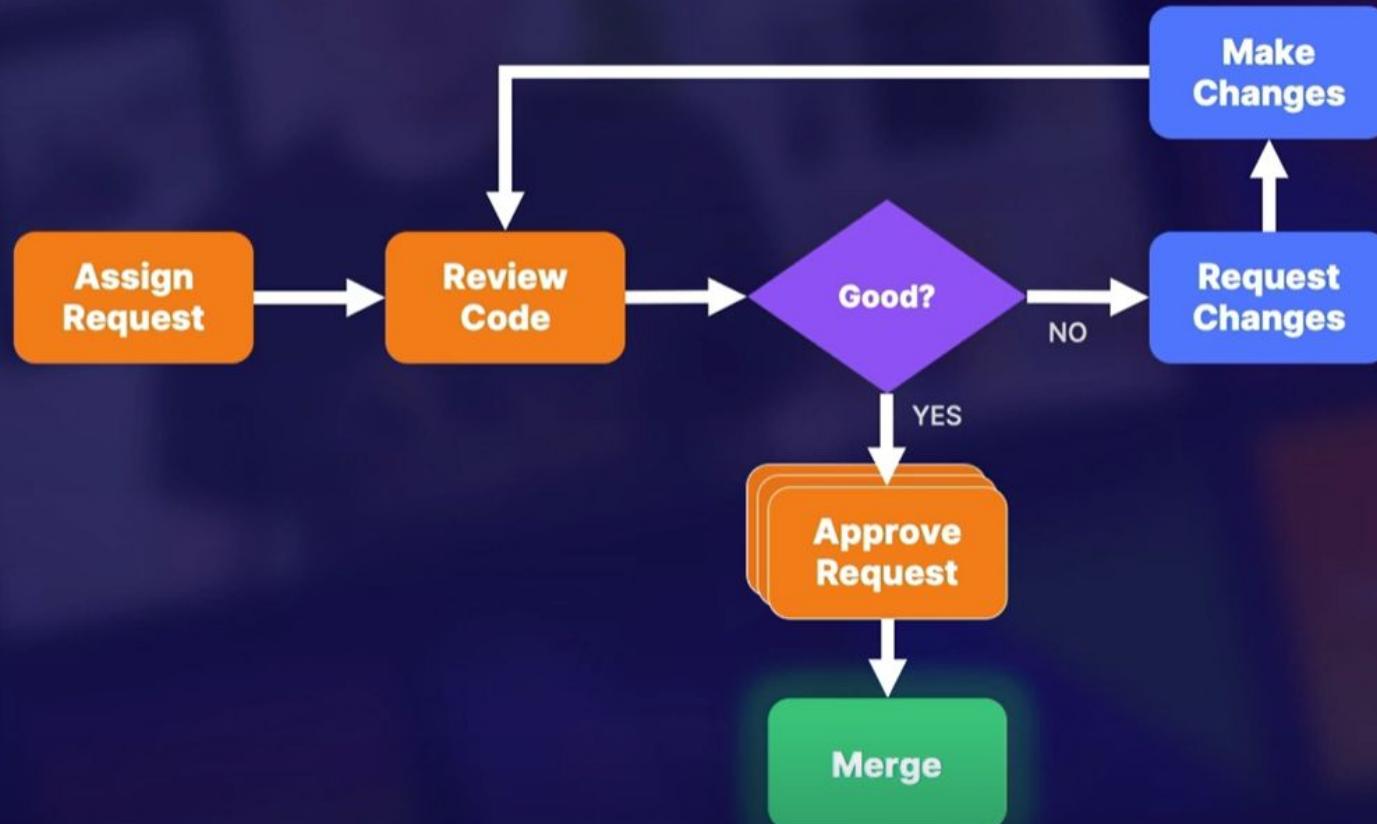
5 References

Work Items, screenshots, links, additional
downloads, or documentation
(validation)

6 The Rest

Challenges, improvements, optimizations,
budget requirements
(other)

Understanding the Pull Request Workflow



Understanding the Pull Request Workflow



Riya
DEVELOPER



Oscar
OPERATIONS



Tom
TEST/QA



Avery
SECURITY

Understanding the Pull Request Workflow



Riya
DEVELOPER

Author

RESPONSIBILITIES

- Ensuring the merge can succeed
- What/Why/How is documented
- Applies suggested code changes

Understanding the Pull Request Workflow



Tom
TEST/QA

Reviewer

RESPONSIBILITIES

- **Review the request details**
- **Review the code***
- **Make comments and suggestions**
- **Approve the code for merging**

Understanding the Pull Request Workflow



Avery
SECURITY

Assignee

RESPONSIBILITIES

- Determine merge timing
- Confirm stakeholder approval
- Delete the old branch

Exploring Code Reviews

Code Review Assignments

1



Automatically choose reviewers.

Scheduled Reminders

2



Send messages to focus attention.

Pull Analytics

3



Metrics help find opportunities!

Exploring Static Code Analysis

Static



Dynamic



Exploring Static Code Analysis

1

Size Limits

Less than 400 lines of code.

Less than 60 minutes of review at a time.

2

Annotations

Authors should guide reviewers.

Provides more in-depth context.

3

Checklists

People make mistakes.

The same ones.

A lot.

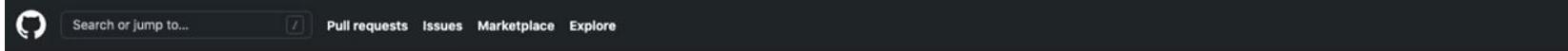
Exploring Static Code Analysis



Code Scanning

- **Coding Errors**
- **Security Vulnerabilities**
- **Find, Triage, Prioritize**

Exploring Static Code Analysis



Marketplace / Apps / DeepSource

Application

DeepSource

[Set up a plan](#)

DeepSource helps you find and fix issues during code reviews. Free to use on open-source repositories.

Seamless integration with GitHub lets you start analyzing code in a couple of minutes. Follow our documentation and guides to get started — deepsource.io/docs/

[Read more...](#)

deepsource

Automate code reviews with static analysis

A screenshot of the DeepSource web interface. The top navigation bar shows 'deepsourcelabs/cli' and 'ANALYSIS ACTIVE'. The main dashboard has tabs for Overview, Issues (selected), Metrics, History, and Settings. Under the Issues tab, it shows 'Recommended' (13) and 'Performance' (4) issues. One specific issue is highlighted: 'Statement not reachable on execution' (Pyl-W0101), categorized as an 'ANTI-PATTERN' in Python, reported a month ago, and is 4 months old. Below the dashboard, there are five cards showing different static code analysis features: 'Automate code reviews with static analysis', 'Get started with DeepSource', 'DeepSource integrates with GitHub', 'DeepSource integrates with Jenkins', and 'DeepSource integrates with CircleCI'.

Exploring Static Code Analysis



Step 2 of 4

Which issues are most important to you?

Configure issues that you would like to see in your pull requests.



Bug Risk

Issues that can cause bugs and breakages in production



Anti-pattern

Code patterns that are correct, but not recommended because they affect maintainability



Performance

Issues that impact performance of the code when it's executed in production



Security

Issues that can potentially be or lead to a security vulnerability



Type Check

Typing violations if you are using a type checker for a dynamically typed language



Coverage

Lapses in test coverage in the source code



Style

Violations in the code format according to a style guide



Documentation

Lapses in source code documentation

[Save issue configuration](#)

Or, [skip for now](#)

Exploring Static Code Analysis

- **Code Analysis Approach:** combination of both Static and Dynamic.
- **Integrate Code Scanning Tools** to automatically test quality and security.
- **Use annotations and checklists** to speed up code reviews and analysis.

Using Pull Requests with Work Items

Main Code



Your Revised Code

Using Pull Requests with Work Items

Main Code



REASON



Work Item

Your Revised Code

Using Pull Requests with Work Items

Main Code



Your Revised Code

REASON



AUDIT TRAIL



Work Item

Using Pull Requests with Work Items

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

The screenshot shows the GitHub interface for opening a pull request. At the top, it displays the base branch as 'main' and the compare branch as 'bugfix-1'. A green checkmark indicates that the branches are 'Able to merge'. The pull request title is 'added bugfix'.

The main body of the PR contains the commit message 'Resolves #1|added information'. Below the message area, there is a placeholder for attachments: 'Attach files by dragging & dropping, selecting or pasting them.' At the bottom right of the PR body is a large green button labeled 'Create pull request'.

To the right of the PR body, there are several configuration sections:

- Reviewers:** No reviews, with a gear icon.
- Assignees:** No one—assign yourself, with a gear icon.
- Labels:** None yet, with a gear icon.
- Projects:** None yet, with a gear icon.
- Milestone:** No milestone, with a gear icon.
- Linked issues:** Use Closing keywords in the description to automatically close issues, with an info icon.

At the bottom left, a note says: 'Remember, contributions to this repository should follow our [GitHub Community Guidelines](#)'.

Using Pull Requests with Work Items



Managing your work with issues and pull requests

- About issues
- Creating an issue
- Deleting an issue
- Opening an issue from a comment
- Opening an issue from code
- Transferring an issue to another repository

- Pinning an issue to your repository

- Creating a permanent link to a code snippet

- Managing labels
- About task lists

- About automation for issues and pull requests with query parameters

- File attachments on issues and pull requests

- Assigning issues and pull requests to other GitHub users

- Viewing all of your issues and pull requests

- Disabling issues

- Linking a pull request to an issue

- About duplicate issues and pull requests

Managing project boards

Tracking the progress of your work with project boards

Tracking the progress of your work with milestones

Linking a pull request to an issue using a keyword

You can link a pull request to an issue by using a supported keyword in the pull request's description or in a commit message (please note that the pull request must be on the default branch).

- close
- closes
- closed
- fix
- fixes
- fixed
- resolve
- resolves
- resolved

The syntax for closing keywords depends on whether the issue is in the same repository as the pull request.

Linked issue	Syntax	Example
Issue in the same repository	<code>KEYWORD #ISSUE-NUMBER</code>	Closes #10
Issue in a different repository	<code>KEYWORD OWNER/REPOSITORY#ISSUE-NUMBER</code>	Fixes octo-org/octo-repo#100
Multiple issues	Use full syntax for each issue	Resolves #10, resolves #123, resolves octo-org/octo-repo#100

Only manually linked pull requests can be manually unlinked. To unlink an issue that you linked using a keyword, you must edit the pull request description to remove the keyword.

You can also use closing keywords in a commit message. The issue will be closed when you merge the commit into the default branch, but the pull request that contains the commit will not be listed as a linked pull request.

Further reading

In this article

- About linked issues and pull requests
- Manually linking a pull request to an issue
- Linking a pull request to an issue using a keyword
- Further reading

Using Pull Requests with Work Items

Settings / Repositories

Search

← My First Test



Filter by keywords

master Default Compare

master

Branch Policies

Note: If any required policy is enabled, this branch cannot be deleted and changes must be made via pull request.



Require a minimum number of reviewers

Require approval from a specified number of reviewers on pull requests.



Check for linked work items

Encourage traceability by checking for linked work items on pull requests.



Required

Block pull requests from being completed unless they have at least one linked work item.



Optional

Warn if there are no linked work items, but allow pull requests to be completed.



Check for comment resolution

Check to see that all comments have been resolved on pull requests.



Limit merge types

Control branch history by limiting the available types of merge when pull requests are completed.

Using Pull Requests with Work Items

- Use “#” to add work item references in a pull request.
- It’s recommended to always correlate work items with pull requests.
- It’s possible to close work items with completed pull requests.

Summary

Configuring Branches

Key Learning Outcomes

Microsoft Exam Expectation:
Implement branch merging restrictions

Definition

- Copy of a codeline
- Help dev teams work together

Key Words

- Commit
- Push
- Fetch
- Pull
- Pull Request
- Merge

Managing

- Policies
- Restrictions
- Protections

Summary

Discovering Branch Strategies

Key Learning Outcomes

Microsoft Exam Expectation:
Define branch strategy

Why You Need It

- Parallel Development
- Structured releases
- Promotion paths
- Quick changes
- Multiple versions

Types

- Trunk-Based
- Feature
- Feature Flag (Toggle)
- Release

Best Uses

- Trunk: really, really quick branches
- Feature: easy CI/CD
- Release: multiple versions

Summary

Understanding the Pull Request Workflow

Key Learning Outcomes

Microsoft Exam Expectation:
Design and implement a PR workflow

Purpose

- Contain info of code changes
- Encourage review and collaboration
- Used before branch merging

What's in it?

- What
- Why
- How
- Tests
- References
- “The Rest”

Ownership

- Author: merge, documentation, applies changes
- Reviewer: comment, suggest, and approve
- Assignee: merge timing and admin tasks

Summary

Exploring Code Reviews

Key Learning Outcomes

Microsoft Exam Expectation:
Define branch strategy

Assignments

- Round Robin
- Least Recent Review Request

Scheduled Reminders

- Messages or other notifications
- Integration to Slack and other tools

Pull Analytics

- Decide what means "good"
- SMART goals
- Find opportunities

Summary

Exploring Static Code Analysis

Key Learning Outcomes

Microsoft Exam Expectation:
Define branch strategy

What is “Static”?

- Code at rest
- Testing code in use is “Dynamic”

Guidelines

- Size limits (400x60)
- Annotations
- Checklists

Tools

- Third-Party Tools
- Find, Triage, Prioritize
- Coding Errors and Security Vulnerabilities

Summary

Using Pull Requests with Work Items

Key Learning Outcomes

Microsoft Exam Expectation:
Define branch strategy

Relationship

- Reasons and more context
- Audit trail for production changes

Adding References

- "#" and the work item number

Enforcing References

- Branch Policies

Summary



Branch Strategies

This is dependency-level knowledge for more advanced-level questions.

Focus on definitions and process.

Configuring Repositories

Configuring Repositories

Configuring Repositories

Using Git Tags to Organize Your Repository

Handling Large Repositories

Exploring Repository Permissions

Removing Repository Data

Recovering Repository Data

Using Git Tags to Organize Your Repository

What are **Git tags**, and why do we care?



Mark specific points in repo history

Notate specific versions – v1.1, v1.2, etc.

Can add notes (annotations) on commit details

“Updated dependencies to version 1.3”

Tags = special ‘name’ applied to a commit

Using Git Tags to Organize Your Repository

Lightweight

- No notes, just a tag name
- Simply a pointer to a specific commit
- Example:

```
git tag v1.2
```

Annotated

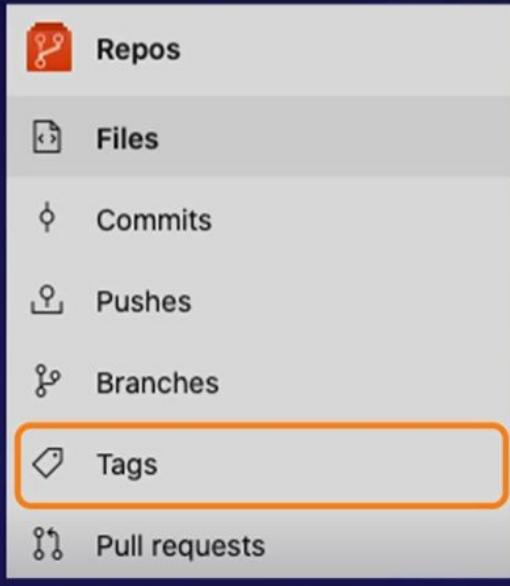
- Attach a note on tag details
- Stored as a full object in Git database
- Example:

```
git tag -a v1.2 -m "Updated  
html template"
```

VS

Using Git Tags to Organize Your Repository

Azure Repos Perspective



Can view and create tags via web portal

Web portal = create annotated tags only

Tags in remote repo require separate push

```
git push origin <tag>
```

```
git push origin v1.2
```

Using Git Tags to Organize Your Repository

The screenshot shows the Azure DevOps interface for a repository named 'customer-website'. The 'Commits' tab is selected in the sidebar. A modal window titled 'Create a tag' is open, prompting for a 'Name' (v1.05), 'Based on' commit (c17d52c4), and a 'Description'. In the background, the commit history for the 'master' branch is visible, showing various commits like 'updated html', 'updated README', and 'Delete .vscode'. A pull request labeled 'v1.1' is also present.

Azure DevOps

ctrl-alt-sweets / customer-website / Repos / Commits / customer-website

Search

customer-website

master

Commits

Graph Commit Pull Request Status

v1.1

Create a tag

Name *

v1.05

Based on

c17d52c4

Description *

Cancel Create

updated html
08084c41

updated README
c17d52c4

Updated README
dfaef9bc1

updated README
2fa36958

Delete .vscode
ca817c57

Update .gitignore
5d8e98c8

Update index.
9fc72c65

Update README
23d1afbd

updated html
8c02e7da

Update index.html
690137ee

Update index.html

Using Git Tags to Organize Your Repository

- **Git tags:** special notes on importance of specific commits
- **Tag types:** lightweight, annotated
- **Azure Repos:** annotated tags only
- **Working with remote repositories:** separate tag push required

Handling Large Repositories

Why do we care about source repository size?

1

Challenge: Git repos are not bulk file storage

- ✓ Small footprint intended
- ✓ Some file types should not be used in repos
- ✓ However, some large binaries must be included

2

Why is Git footprint size important?

- ✓ Cloning repo copies **full** history of all file versions
- ✓ Frequently updated large files = serious performance issues!



Handling Large Repositories

Solution(s)

- 1 **Use best practices to avoid size 'bloat'.**
 - ✓ Know what **not** to include.
- 2 **For unavoidable large files, use Git LFS.**
- 3 **Clean up accumulated 'clutter' with git gc**



Handling Large Repositories

Git LFS

1 Large file management built into Git

- ✓ Open-source extension (separately installed)
- ✓ Supported by popular remote repos (GitHub, Azure Repos)
 - Tagged large files stored by remote repo, but as a placeholder in actual source code



Handling Large Repositories

1 Install Git LFS for your OS.

2 Initiate LFS on your local environment.

```
git lfs install
```

3 Tag files to be added to LFS before committing them.



This results in a `.gitattributes` file.

```
git lfs track "*.psd"
```

4 Commit as usual. The remote repo will store tagged files separately.



Tagged file will be marked in line with source code.

Handling Large Repositories

Do not commit...

Frequently Updated Binaries

With LFS: intended for infrequent updates.

Without LFS: leads to multiple copies in history.

Code Outputs

Outputs do not belong in source code.

Azure Boards can be used to better track work items.

Dependencies

Use package management solutions instead (discussed later).

Handling Large Repositories

1 git gc = 'garbage collection'

- ✓ Loose files in unreferenced commits
- ✓ Large binaries

```
git gc --(option)
```

2 git gc + option compresses large files and/or removes loose files

3 git gc options

- ✓ --aggressive – Full optimization/compression (takes longer to run)
- ✓ --prune – prune (delete) all loose objects --prune=<date> – prune (delete) loose objects older than <date>
- ✓ --no-prune – do not prune any loose objects

Handling Large Repositories

- ✓ **Avoid large file 'bloat':** Large files in history drag down performance
- ✓ **Git LFS:** Open-source large file management | Remote repo markers
- ✓ **Best practices:** Keep unnecessary large files out of source code – alternative solutions
- ✓ **git gc:** Garbage collection – know flags to keep/prune loose files

Exploring Repository Permissions

Branch Permissions in Azure Repos

1

Branch-level permission access

- ✓ Provides users different access to different branches
- ✓ By default: inherited from organization/project-level roles
- ✓ Provides access to main branch but not sub-branch, or vice versa



Exploring Repository Permissions

How Branch Permissions Work

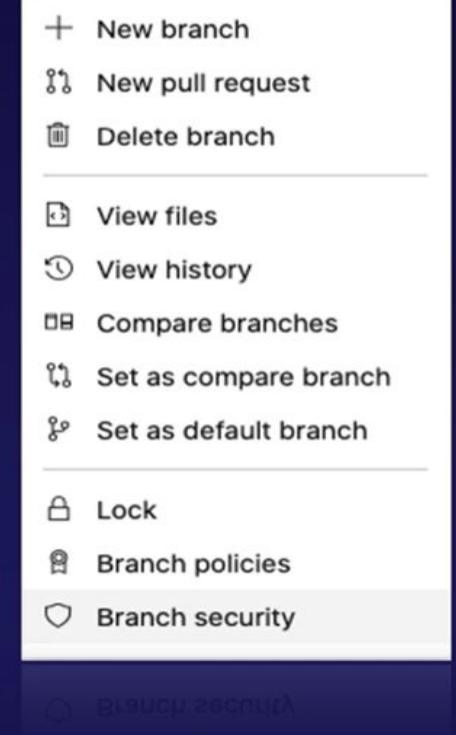
1 Access the **Branches** menu from the web portal.

2 From the context menu, select **Branch security**.

3 Assign **permissions** to users.

✓ Permissions are inherited from the project/organization level by default.

✓ You can explicitly deny permissions granted by higher-level roles.



Exploring Repository Permissions

Branch Locks

Lock branch in 'read-only' mode

No new commits

Cannot change existing commit history

Purpose: Pair with pull requests

Prevent changes while being reviewed

Removing branch locks

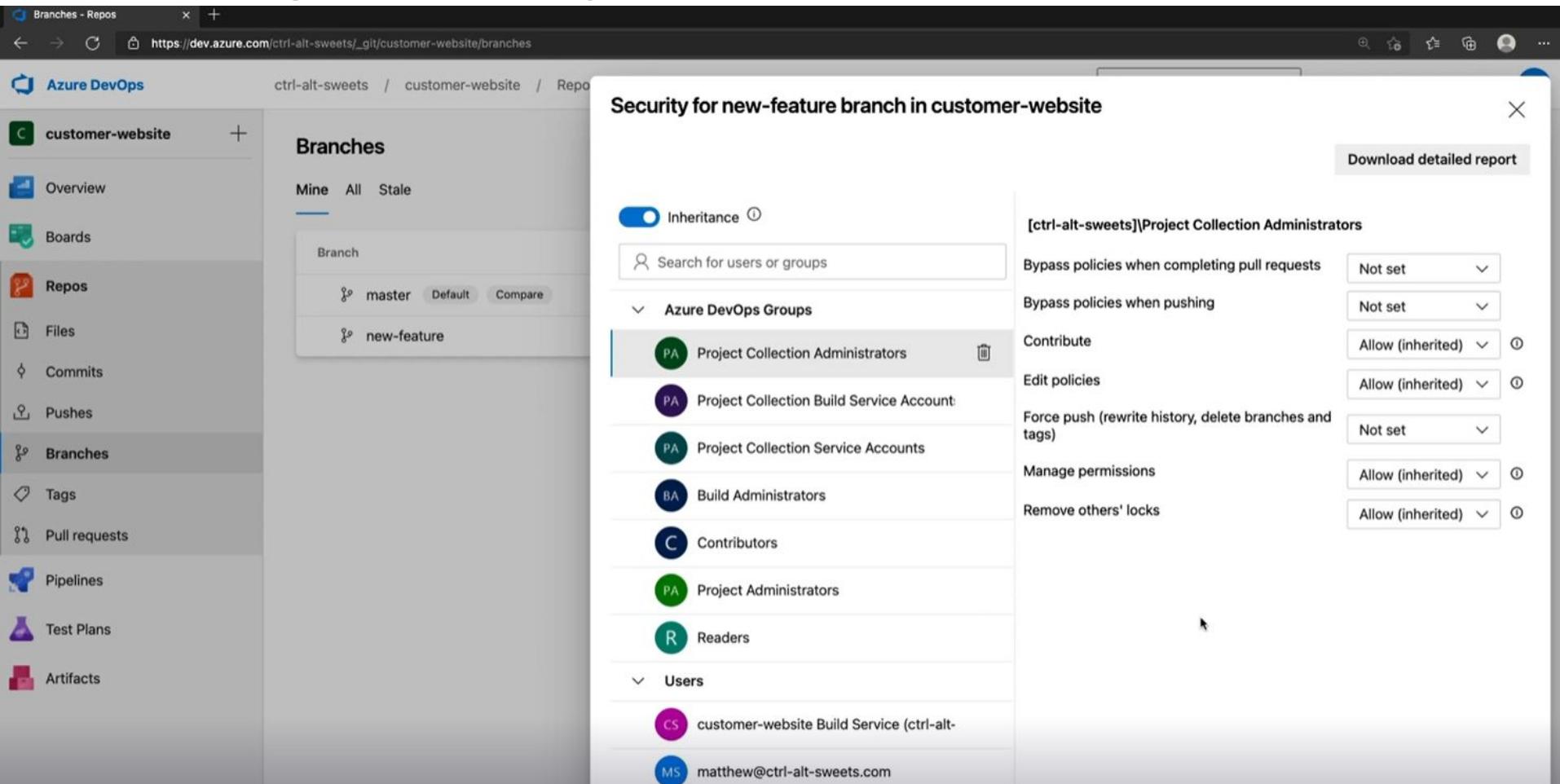
User who locked it

Project Administrator

Explicit 'remove lock' permission assigned



Exploring Repository Permissions



The screenshot shows the Azure DevOps interface for managing repository permissions. On the left, the sidebar navigation includes: Overview, Boards, Repos (selected), Files, Commits, Pushes, Branches (selected), Tags, Pull requests, Pipelines, Test Plans, and Artifacts. The main area displays the 'customer-website' repository's branches: master (Default) and new-feature.

A modal window titled 'Security for new-feature branch in customer-website' is open, showing the current security settings for this specific branch. The 'Inheritance' toggle is turned on. The 'Azure DevOps Groups' section lists several groups with their respective permission levels:

- [ctrl-alt-sweets]\Project Collection Administrators: Bypass policies when completing pull requests (Not set), Bypass policies when pushing (Not set), Contribute (Allow (inherited)), Edit policies (Allow (inherited)), Force push (rewrite history, delete branches and tags) (Not set), Manage permissions (Allow (inherited)), Remove others' locks (Allow (inherited)).
- Project Collection Build Service Account: (No specific permissions listed)
- Project Collection Service Accounts: (No specific permissions listed)
- Build Administrators: (No specific permissions listed)
- Contributors: (No specific permissions listed)
- Project Administrators: (No specific permissions listed)
- Readers: (No specific permissions listed)

The 'Users' section lists two individuals:

- customer-website Build Service (ctrl-alt-sweets)\customer-website Build Service (MS): (No specific permissions listed)
- matthew@ctrl-alt-sweets.com (MS): (No specific permissions listed)

At the top right of the modal, there is a 'Download detailed report' button.

Exploring Repository Permissions

The screenshot shows the Azure DevOps interface for a repository named 'customer-website'. The left sidebar has 'Branches' selected. The main content area displays two branches: 'master' and 'new-feature'. The 'new-feature' branch is currently selected. A tooltip indicates it is 'Locked by matthew@ctrl-alt-sweets.com'. The table headers are 'Branch', 'Commit', 'Author', 'Author...', 'Behind | Ahead', 'Status', and 'Pull Re...'. The 'new-feature' row shows commit '08084c41' by 'M...' on 'Tuesday'.

Branch	Commit	Author	Author...	Behind Ahead	Status	Pull Re...
master	08084c41	M...	Tuesday			
new-feature	08084c41	M...	Tuesday	0 0		

Locked by matthew@ctrl-alt-sweets.com

Exploring Repository Permissions

- ✓ **Main takeaway:** Ability to manage branch-level permissions in Azure Repos
- ✓ **Inheritance:** Pull from organization/project groups | can add/override inherited roles
- ✓ **Branch locks:** Lock a branch in 'read-only' mode for pull requests

Removing Repository Data

Unwanted Files in Git

Scenario: You mistakenly commit and/or push files that you should not include.

- ✓ Very large files
- ✓ Sensitive data (passwords, SSH keys, secrets)

Problem: Deleted files from new commit still exist in the repo's history.

- ✓ Still discoverable by searching through history

Solution: Remove 'bad' commits before or after pushing to a remote repo.

- ✓ Can also remove files from history (with caveats)



Removing Repository Data

Local Commit but Not Yet Pushed

- 'Bad' commit on local environment but not yet pushed to remote repo
- Solution: Remove/amend bad local commit

Pushed to Remote Repo

- 'Bad' commit pushed to remote repo
- Solution: Delete remote commit
- Alternatively: Remove unwanted file history...with caveats

VS

Removing Repository Data

Removing/Amending Local Commit Before Push

- 1 Delete unwanted file.
- 2 Remove file from Git tree/index.
`git rm --cached <filename>`
- 3 Delete or amend previous commit (depending on what other data changed).

Entirely Delete Commit

```
git reset HEAD^
```

Amend Commit

```
git commit --amend -m "comment"
```

Removing Repository Data

Remove Commit Already Pushed to Remote Repo

Best for recent pushes

- 1 Reset back to last 'good' commit.

```
git reset --hard #commitSHA
```

- 2 Force push to remove commits past the last good one.

```
git push --force
```

- 3 All branch commits past the reset one will be deleted.

Removing Repository Data

Remove Unwanted File from Past Commits' History

Short version: no single best solution

- 1 There are multiple tools to remove files from past history. Some are 'official', others are community-created.

- ✓ **git filter-branch**: built-in method
- ✓ **git filter-repo**: officially recommended community solution
- ✓ **BFG Repo Cleaner**

- 2 Even after successful removal, you can still view file history in the Azure Repos web portal.

Recovering Repository Data

Mistakes happen!

Scenario: Changes made/deleted in error

- ✓ Pushed commits containing errors
- ✓ Mistakenly deleted a branch in Azure Repos
- ✓ Mistakenly deleted entire Azure repository



Need to know how to recover or 'rewind time' in the above scenarios

Recovering Repository Data

Scenario: Commits contain errors — need to 'roll back'

Same process as previous 'reset commits' lesson

- 1 Reset back to last 'good' commit, and resume development from there.

```
git reset --hard #commitSHA
```

- 2 Coordinate with development team members to merge changes to reverted code.



Known as 'rebase'

Recovering Repository Data

Scenario: Restore deleted Azure Repos branch

1

In Azure Repos, from the Branches view, search for the deleted branch.

2

From the deleted branches search, click Restore branch.

A screenshot of the Azure Repos Branches view. The title bar says "Branches". Below it are three filter buttons: "Mine", "All", and "Stale". A search bar contains the text "users/raisa/css-updates". To the right of the search bar is a red rectangular box containing the text "Search for exact match in deleted branches". Below the search bar, a message says "No branches match the filter: users/raisa/css-updates.".

A screenshot of the Azure Repos Branches view. The title bar says "Branches". Below it are three filter buttons: "Mine", "All", and "Stale". A message says "Exact match found". Below that, a tree view shows a folder "users/raisa" expanded, revealing a branch "css-updates". To the right of the branch name is a red rectangular box containing three dots (...). At the bottom right is a button labeled "Restore branch" with a red rectangular box around its icon.

Recovering Repository Data

Scenario: Restore entire deleted Azure repository

1

Despite warning, repo is in a 'soft delete' state and can be restored.

2

Restore via an authenticated API call.

PATCH

`https://dev.azure.com/{organization}/{project}/_apis/git/recycleBin/repositories/{repositoryId}?api-version=6.0-preview.1`



Thank YOU