Product KPIs (PKPIs)

This document describes product KPIs that can be used to determine the 'health' of the overall product and its constituent repositories.

Goals of PKPIs

Traditional KPIs used by the engineering services team measure metrics like service reliability, service availability, queue wait times, etc. Analysis of the indicators (e.g. against a baseline expected value) helps the engineering team determine whether action needs to be taken to ensure that the product can build and ship on time. For example, a spike in Helix queue times will affect how fast PRs can be validated, which in turn affects how fast the product can iterate.

Though product KPIs bear limited resemblance to traditional service KPIs, the goal is largely the same:

Ensure that the product can iterate and ship quickly with high quality by identifying problem areas early and often.

As a quantitative goal, the PKPI's defined here help drive towards the goal of being able to produce a product build in under 2 hours.

Audience

There are two main audiences for these PKPI's. **The primary audience is the repo owners.** The secondary audience is our leadership team (Joc, Steve, etc…). The audience should be kept forefront in mind when determining an implementation.

Scope

The PKPI's will be applicable to all channels, but primarily focused on .NET 5.

Metrics

The desired product KPI metrics are listed below, with the following fields:

- **Name** - Name of metric
- **Qualifier** - How is this metric broken out? For example, dependency update PR merge time broken out by input channel, or are all dependency update PRs aggregated per repo. Is this metric an aggregator of all repos underneath it? For example, staleness can be aggregated such that even if all direct inputs to a repository A are new, if those inputs have stale inputs, then A is viewed as stale.

- **Drives Action** - What action can be taken to improve the metric and thus improve the chances of shipping with high quality.

Name	Workstream	UI	Qualifier	Indicates	Drives Action	Notes
Longest Official Build Time Path		PowerBI	For default output channel	Aggregation of constituent build times including dependency flow times (SKPI, not PKPI) and mirror times (SKPI, not PKPI).	Drives secondary audience (Jared, Steve, dotnetes) to monitor and ensure “good” time.	

Name	Workstream	UI	Qualifier	Indicates	Drives Action	Notes
Official Build Time	Official build	PowerBI	per repo, Per default output channel	Measures the median build time of *successful* official builds of a branch that are applied by default to the specified channel. This should include time for retries. A large official build time indicates that it is slow to move dependencies through the node or get new outputs from the node, and that failures in the official build that require retries will be expensive, potentially putting ship schedules at risk. chcosta - task time and trend time	Repository owner investigates official build time, determines why it has regressed, and develops a plan to mitigate it.	Epsitha

Name	Workstream	UI	Qualifier	Indicates	Drives Action	Notes
Official Build Pass Rate	Official build	PowerBI	per repo, Per default output channel	Measures the average 'without intervention' pass rate over the last N days of builds of a branch that are applied by default to the specified channel. A build that is retried after an initial failure counts as a failure. A low pass rate indicates that dependencies or new content will not flow easily through the node, potentially putting ship schedules at risk.	Repository owner investigates official build pass rate, determines why it has regressed, and develops a plan to mitigate it.	Epsitha

Name	Workstream	UI	Qualifier	Indicates	Drives Action	Notes
Percent of changes that don't require a dependency Update PR	Dependency updates	PowerBI	per repo, Per default output channel	Indicates a failure in the build that requires a fix, fragility between components, or poor official build pass rates.	Repository owner determines why dependency update PRs are required and investigates how to have cleaner dependency flows going forward.	Megan- Ideally this number should be zero.- Not actionable until new dependency flow is in place
Dependency updates that flow seamlessly	Dependency updates	PowerBI	per repo, per output channel	Contributes to answering the question "how many times a day to dependencies flow through the system". Indicates healthy dependency flow.		Michelle

Name	Workstream	UI	Qualifier	Indicates	Drives Action	Notes
Dependency updates that fail, open a corresponding PR, and that PR fails initially	Dependency updates	PowerBi	per repo, per output channel	Contributes to answering the question “how many times a day do dependencies flow through the system”? Indicates the officaial build detected a break and that break was confirmed by PR testing	Repository owner investigates to understand why dependencies are not flowing	Michelle
Dependency updates that fail, open a corresponding PR, and that PR fails instantly	Dependency updates	PowerBi	per repo, per output channel	Contributes to answer the question “how many times a day do dependencies flow through the system”? Indicates a merge conflict that requires manual intervention.	Repository owner investigates to understand why dependencies are not flowing (merge conflict)	Michelle

Name	Workstream	UI	Qualifier	Indicates	Drives Action	Notes
Dependency updates that fail, open a corresponding PR, and that PR passes and is auto-merged	Dependency updates	PowerBi	per repo, per output channel	Contributes to answering the question "how many times a day do dependencies flow through the system"? Indicates official build detected a break, but that break did not show up in PR testing.	Highlights a gap in PR testing that may need to be addressed by the repo owner.	
Dependency updates that require a merge commit	Dependency updates	PowerBi	per repo, per output channel	Contributes to answering the question "how many times a day do dependencies flow through the system"?		

Name	Workstream	UI	Qualifier	Indicates	Drives Action	Notes
Number of dependency updates per given time frame	Dependency updates	PowerBi	per output channel, per repo	Measures rate at which dependency updates are produced. Lower numbers are better, large numbers indicate unneccessary complexity. As product moves towards shipping, number of dependency update PR's decreases.	Repository owner investigates open dependency updates and determines if increasing or current numbers are valid.	Megan- This is a pulse metric, giving a sense of product health but may not be specifically actionable by repo owners.

Name	Workstream	UI	Qualifier	Indicates	Drives Action	Notes
Direct Product Dependency Staleness	Dependency staleness	BarViz	per output channel, per repo	Measures the staleness of the direct *product* inputs of a repository, ignoring pinned dependencies. A dependency's staleness is measured against the latest version of the dependency applied to the input channel.For example, if the repository has dependency 'Foo.Bar', and 'Foo.Bar' is updated with a subscription to repo 'A' on channel 'Dev', then the dependency is not considered stale if repo A has produced no new builds containing Foo.Bar on channel Dev.A	Repository owner finds dependency update PRs for old dependencies and fixes/merges as needed. If the PRs do not exist, the repository owner investigates why (missing subscription, etc.)	

Name	Workstream	UI	Qualifier	Indicates	Drives Action	Notes
Direct Toolset Dependency Staleness	Dependency staleness	BarViz	per output channel, per repo	Measures the staleness of the direct *toolset* inputs of a repository, ignoring pinned dependencies. A dependency's staleness is measured against the latest version of the dependency applied to the input channel.For example, if the repository has dependency 'Foo.Bar', and 'Foo.Bar' is updated with a subscription to repo 'A' on channel 'Dev', then the dependency is not considered stale if repo A has produced no new builds containing Foo.Bar on channel Dev.A	Repository owner finds dependency update PRs for old dependencies and fixes/merges as needed. If the PRs do not exist, the repository owner investigates why (missing subscription, etc.)	

Name	Workstream	UI	Qualifier	Indicates	Drives Action	Notes
Existence of Product Dependency Cycles	Dependency flow	Darc / BarViz	Per Channel	Indicates the existence/non-existence of cycles containing only product dependencies within the dependency flow graph. A cycle unbroken by toolset a toolset dependency indicates the product cannot be become coherent.	Coherency QB or repository owner that is part of the cycle investigates and determines how to break the cycle	- this may already be done for darc

Name	Workstream	UI	Qualifier	Indicates	Drives Action	Notes
Existence of Automated dependency Flow Cycles	Dependency flow	Darc / BarViz	Per Channel	Indicates the existence/non-existence of cycles within the dependency flow graph that will flow without intervention. A subscription will flow without intervention if it is not disabled and its frequency is not 'none'. This metric is a shorthand way of saying that the product will never become coherent if 'real' product changes in the repositories are halted, potentially putting ship schedules at risk. In Dev channels, this is not an interesting metric.	For a release channel, the coherency QB and/or repository owner determines which subscriptions in the cycle do not need to flow automatically, and disables/changes update frequency as needed.	

Name	Workstream	UI	Qualifier	Indicates	Drives Action	Notes
Existence of cross-channel flow	Dependency flow	Darc / BarViz	Per Channel	Indicates the existence/non-existence of flow that may enter the specified channel from another channel. If the other channel has a higher rate of change than the specified channel, then the product may not end up becoming coherent if left to its own devices. A good example of this is: A branch that applies to Release has a subscription to Arcade (on .NET Tools - Latest) that is set to update every day. Because branches applying to the release channel and arcade's	For a release channel, the coherency QB and/or repository owner determines which repositories are contributing to the cross channel flow (using Darc/BARViz) and determines a plan of action (e.g. branch offending repos, turn off automated subscriptions contributing to the flow, etc.)	

Name	Workstream	UI	Qualifier	Indicates	Drives Action	Notes
Missing/Disabled Product Input Subscriptions	Subscriptions	Darc / BarViz	per repo, Per Default Channel (input branch)	Indicates that a branch that applies to a default channel is missing inputs subscriptions for product dependencies, and thus coherency may not be able to be achieved. A subscription is not missing when dependencies are pinned or when they are tied via CoherentParentDependency attributes to other dependencies. Subscriptions that do not update automatically should also be viewed as disabled. Each dependency should be evaluated based on the assets produced	chcostaRepository owner determines whether those input dependencies need to be marked as Pinned or whether subscriptions should be created/changed to update automatically	

Name	Workstream	UI	Qualifier	Indicates	Drives Action	Notes
Missing/Disabled Toolset Input Subscriptions	Subscriptions	Darc / BarViz	per repo, Per Default Channel (input branch)	Indicates that a branch that applies to a default channel is missing inputs subscriptions for toolset dependencies. A subscription is not missing when dependencies are pinned or when they are tied via CoherentParentDependency attributes to other dependencies. Each dependency should be evaluated based on the assets produced by the last build applied to the input subscriptions. This metric is inexact. It's possible that a dependency missing a subscription now will be produced	Repository owner determines whether those input dependencies need to be marked as Pinned or whether subscriptions should be created/changed to update automatically. In a release channel, this is not generally needed but should be evaluated on a case-by-case basis.	chcosta- This is already done in darc, needs to be done in barviz

Name	Workstream	UI	Qualifier	Indicates	Drives Action	Notes
Superfluous Input Subscriptions	Subscriptions	Darc / BarViz	per repo, Per Default Channel (input branch)	Indicates that a branch that applies to a default channel has subscriptions that may do nothing. Each dependency should be evaluated based on the assets produced by the last build applied to the input subscriptions. This metric is inexact. It's possible that a subscription that looks superfluous now will produce a needed asset in the future.	Repository owner determines whether those subscriptions can be deleted.	chcosta- This is already done in darc, needs to be done in barviz

Name	Workstream	UI	Qualifier	Indicates	Drives Action	Notes
Conflicting Input Subscriptions	Subscriptions	Darc / BarViz	per repo, Per Default Channel (input branch) (darc / barviz)	Indicates that a branch that applies to a default channel has subscriptions that produce the same asset. Each dependency should be evaluated based on the assets produced by the last build applied to the input subscriptions. This metric is inexact. It's possible that two currently conflicting subscriptions will not produce conflicting outputs in the future. Because outputs switching between repositories happens infrequently, this metric typically indicates an actual	Repository owner determines whether both subscriptions are needed or whether one should be deleted.	chcosta- this might already be done

Was this helpful? 👍 👎