Document status: WIP | In Review | Approved | **Retired**

This document has been created to focus on, and possibly make a significant iteration on, the table in the current Platform Maturity Model paper without removing those comments in the process. Please refer back to the current paper for any additional context: https://docs.google.com/document/d/1bP8-LQ-d41eIdQB3IC2YsncDhawpFLggql2JxwtE0XI/edit?usp=drivesdk

Also maybe useful to review is the group brainstorm work here: https://drive.google.com/drive/folders/17sJUxSGcsCtUBMUdrwXipk9WnUb3_QmI

As you review:

Please keep in mind the following principles that we agreed are important
(this is also open for comment!):

- This is a model for platform providers, we are not focusing on other personas
- Our target audience is platform providers at the architect/director/lead level. That means not mid level engineers, nor CTOs of multi national organizations.
- Avoid overlapping between characteristics/aspects/rows
- Aim for progression not only across the levels (left to right), but also across the aspects (top to bottom)
- Readability / consumability is key, this can relate to the breadth and depth of the table (keep number of rows/columns low, and number of words per box low)
- Level naming is helpful, but keep in mind:
    - All levels should be reasonable stopping points (therefore positive wording)
    - Still provide numbering for easy discussion
- We value asking questions to clarify the intent of each row
- We value clarifying the movement between levels (and not just the levels themselves)
- Focus on outcomes over implementations with the goal of self learning
- Platforms are inherently socio-technical and this model should reflect that

Also keep in mind these open questions:

- Naming is hard! Keep refining all names.
- The platform maturity model is different from the cloud native one, but how do we intend to fit into that model and what level of alignment is necessary / desirable?

Also, remember that this is the snapshot view and we intend to detail a lot more about each of these boxes in the published paper. With this in mind, **the table should not be confusing or misleading, but feelings of curiosity or incompleteness is inevitable**. The clarifying column was introduced to try and reduce the concerns, so if you find yourself agreeing in principle with the intention and the words but wanting to make sure a certain nuance is included in the description, please hold off on that nuance until the table is settled and we dive into those details.

And finally, thank you for reviewing!

				...requires *gathering* and *consolidating* to get to...	...requires *structuring* and *scaling* to get to...	...requires *collaborating* and *adapting* to get to...
		(this column is only to make review easier and this type of content will be moved to a more detailed section in the future)	**1: Provisional**	**2: Operationalized**	**3: Scaled**	**4: Optimized**
How does platform engineering work get prioritized and financed?	**Investment**	Should capture the company investment of both time and money into platform efforts. *Touched on white paper attribute(s): platform as a product*	Voluntary or temporary	Technical cost center	Product budget	Profit center
How are users discovering and integrating with the platform capabilities?	**Adoption**	Should focus on the “optional” principle mentioned in the white paper. Also that some efforts will be top down and others bottom up and hopefully we can support both. *Touched on white paper attribute(s): user experience, documentation and onboarding, optional*	Erratic	Incentives or Directives	Provider driven	Community enabled
How do users interact with and consume platform capabilities?	**Interface**	Focused on the goal of reducing cognitive load on users as well as making offerings scale better than linearly. *Touched on white paper attribute(s): user experience, onboarding, self-service, reduced cognitive load*	Bespoke processes	Supported solutions	Self-service solutions	API contracts
Once a capability is in use, how is its lifecycle managed including upgrades and EOL?	**Operations**	Should tackle the hard problem of “day 2” for platform capabilities and how some models leave things like security patching at risk due to an unclear ownership model. *Touched on white paper attribute(s): platform as a product, documentation and onboarding, reduced cognitive load for users, secure by default*	By request or requirement	Centrally tracked	Centralized management	Defined responsibility
What is the process for gathering and incorporating feedback and learning?	**Measurement**	This should focus on the feedback loops that the engineers enable. It should highlight the socio-technical side of measurement and not just be about data collection, but also data use. *Touched on white paper attribute(s): platform as a product, optional*	Ad hoc	Consistent collection creating prioritized tasks	Quantitative and qualitative driving high level objectives	Multiple tiers influencing all levels of platform mission

Notes for the detailed sections:

- General format:
  - Each aspect should have a high level summary of its value (small paragraph)
  - Each aspect should then have details per level
  - Level details within an aspect should be no more than a few sentences
  - Level details can include practices / techniques (but not projects / tools)
  - Level details can refer to external, publicly available resources that reinforce

the details
    - It is OK (even expected) that some of the things discussed will be only mentioned and could benefit from a much deeper dive. This is out of scope for this paper and is an opportunity to add a GitHub Issue to track a deep dive paper on the subject.

- For “adoption”, we may want to add some context around easier ways to get started during the provisional or operationalized levels. As in, a look at how managing ops style work at this stage may be easiest (but at the same time not an indication of maturity to start there). See comment for context: Table deep dive

- For interface, be conscious that “solutions” is intentionally vague since we may not want to indicate that automation is the only way to succeed. Now of course at scaled speaking to automation will be key, but actually a very well documented process that doesn’t require any hand holding or human wait times is scaled!

- Extensibility, as in how non-platform owners can add to the features of the platform, should be captured throughout a few different aspects in the detailed section. Namely it was discussed under both investment and interface (and maybe adoption at the optimized level?).
    - The levels we discussed was the difference between closed, limited, plugin or complete developer ecosystem (with inner or open sourcing)

- For “community enabled” under adoption, this can also be described as the flywheel. Where people are using the platform, contributing back to it, and that drives more engagement.
    - A mature organization should do inner-sourcing and have their developers contribute back to the platform to make it more relatable to their use case. Plus they also become the advocate/ambassador of the platform. Companies could have internal ambassador programs at a later stage too.

- Under “measurement”, this should raise that different platforms are targeting different values. One may be developer productivity. And whatever the goals they should be measured (good luck measuring dev productivity 😂)
    - Also review the need to focus on the process for feedback collection in the details section. Please see comment for a lot of ideas: Table deep dive

- For “operations” it is a bit harder to parse at the table level and less obvious that it goes in a progressive way.
    - This should capture the maintenance of running resources (things actually provided by the platform). And also running and supporting it.
    - For “optimized” it is important to address the fact that collaboration is healthy, but clarity on the responsibility to complete something is clear.
    - A term to use could be “decentralized” or “distributed” as a way to capture when a platform exposes responsibility clearly to its users.
    - For “scaled” should evoke the idea that when the platform has created a number of instances of a capability (e.g. many databases), they should be

updateable as a collective rather than as independent items. Some conventionality and consistency around the process.

- For “optimized” an implementation should be a shared responsibility model as used by the cloud providers to indicate who owns what for (in their example) security.