- Name of the project (existing or proposed):

Rez

- Requested project maturity level (select one): Adopted or Incubation

Incubation

- Project description (please describe the purpose and function of the project, its origin and its significance to the ecosystem):

Rez is a cross-platform package manager. Using Rez you can create runtimes configured for a given set of packages. However, unlike many other package managers, packages are not installed into these runtimes. Instead, all package artifacts are installed into a central repository, and runtimes reference these existing artifacts. As a consequence, runtimes have a minimal footprint can be constructed quickly, often within a few seconds.

The first version of rez was implemented at Dr.D studios in Sydney, and was called drd-config (it was initially close-sourced, but was made open source around the same time the company was shelved). The original motivation for the project came from having worked at MPC London previously, where we'd started running up against significant issues related to package management, that at the time we did not have a suitable solution for.

Rez is used widely in the VFX industry as its ability to configure runtimes with minimal overhead makes it useful for managing a large number of different configurations at once (corresponding to the large number of active shows and tools common within many studios in our industry).

- Please explain how this project is aligned with the mission of ASWF?

The intent of rez from the beginning was to provide a common platform for package management in the VFX industry, which I think quite closely aligns with the ASWF's mission to "provide a common build [and test] infrastructure" in VFX. Rez aims to solve a problem common to virtually every studio, and unsurprisingly we do see a number of similar, closed-source solutions in place in various studios. I believe that by collaborating on a common project instead, we can minimise redundant efforts at solving this problem; help share knowledge of one system across studios so that expertise is more easily accessible; help share software interstudio if and when that's required; and overall minimise the internal resources a given studio needs to put towards managing their software.

- What is the project's license for code contributions and methodology for code contributions. ASWF maintains recommendations for contribution and licensing for hosted projects.
    - License is (as of very recently) Apache 2.0.
    - LICENSE file is included in root of project
    - Copyright and license header is included in every source file
    - CONTRIBUTING.md present
    - DCO signoffs are not present
    - CLA is not in use
    - No code of conduct provided

- What tool or platform is utilized for source control (GitHub, etc.) and what is the location (e.g. URL)?

https://github.com/nerdvegas/rez

- What are the external dependencies of the project, and what are the licenses of those dependencies?

See https://github.com/nerdvegas/rez/blob/master/src/rez/vendor/README.md

- What roles does the project have (e.g. maintainers, committers?) Who are the current core committers of the project, or which can a list of committers be found?

I would have to describe myself as the sole maintainer (I'm the only one who merges to master and manages releases), however there are significant contributions from other members.

Committers: https://github.com/nerdvegas/rez/graphs/contributors

- What mailing lists are currently used by the project?
    - Slack (rez-talk)
    - https://groups.google.com/g/rez-config

- What tool or platform is leveraged by the project for issue tracking?

GitHub.

- Does the project have a Core Infrastructure Initiative security best practices badge? Do you foresee any challenges obtaining one? (See: https://bestpractices.coreinfrastructure.org)

No it does not have the badge.

I don't see challenges in obtaining one, as it seems that most requirements are already met. However points that may need to be addressed are summarized below (taken from https://bestpractices.coreinfrastructure.org/en/criteria/0):

- The project website MUST provide information on how to: obtain, provide feedback (as bug reports or enhancements), and contribute to the software. *(minor updates to CONTRIBUTING.md required)*
- The project MUST provide reference documentation that describes the external interface [...] *(Sphinx docgen is provided, however public API is not well defined, this will need some work)*
- To enable collaborative review, the project's source repository MUST include interim versions for review between releases *(Currently we do not have release candidates, only standard releases)*
- The project MUST publish the process for reporting vulnerabilities on the project site *(honestly I'm not clear on what vulnerabilities mean in the context of rez presently)*
- It is SUGGESTED that the test suite cover most (or ideally all) the code branches, input fields, and functionality *(testing is good in general but more could be done to establish actual coverage)*
- The project MUST have at least one primary developer who knows how to design secure software *(I don't know what this means in practice tbh)*
- At least one of the project's primary developers MUST know of common kinds of errors that lead to vulnerabilities in this kind of software, as well as at least one method to counter or mitigate each of them *(see above)*
- At least one static code analysis tool (beyond compiler warnings and "safe" language modes) MUST be applied to any proposed major production release of the software [..] *(we do not currently have any form of static analysis in place)*
- It is SUGGESTED that at least one dynamic analysis tool be applied [...] *(what is dynamic analysis and how is that different to a test suite?)*

- What is the project's website? Is there a wiki?

https://github.com/nerdvegas/rez/wiki

- What social media accounts are used by the project?

None.

- What is the project's release methodology and cadence?

Release methodology is basic and should be updated. Currently we simply perform a linear set of semver releases (patch or minor - major version changes are extremely uncommon). We do not have formal releases where some amount of production testing has been quantifiably done beforehand.

Cadence: 66 releases over the past year, so just over one release a week on average. All releases are summarized here:
https://github.com/nerdvegas/rez/blob/master/CHANGELOG.md

- Are any trademarks, registered or unregistered, leveraged by the project? Have any trademark registrations been filed by the project or any third party anywhere in the world?

No.