1. [5/31/2019]

... Should I Use `circle.yml` or Add A Hook Script?

In Canaveral, each "pipeline" segment is made up of one or more systems that can do that job (and often several that can do parts of the job). For instance, the "src" segment uses GitHub while the build segment uses CCI (or Jenkins if you're brave). Each of those segments has standards to how it does things, so for instance, GitHub has some semantics to how you use it, and CCI has the circle.yml file.

The original view of Canaveral was that we would *control* all of those things mostly because we were primitive and didn't want to foist that off onto customers. As we have evolved, we have been relaxing that requirement. For instance, instead of controlling the `circle.yml` file we allow users to edit it as needed.

The `circle.yml` contents are specific to CCI so that's why you see a `blueprint.yml` to talk with Canaveral Deploy Engine and a `.github/contribs.txt` if you use that feature, etc. What you see is really more **based on what features you use and which you don't**. The contents of those files are initially laid out by Canaveral, but we have a very strict **"your code, your control"** model.

You can modify the `circle.yml` (no system will complain); however, in modifying that file a few things change. For instance, modifying the file will mean any automated update systems we have (which automatically address issues in that file) can't fingerprint match on it anymore, and so will mark it as **"customized"** and skip it.

> We have had to do this a few times, **the next time will be the move to CCI 2.0**. In those cases, it will be **your responsibility** to make the correct changes for your workflow.

In order to try and support **"both worlds"** most of our systems support hooks at various points to allow you to "plugin" to the system without changing our intended logic. In this case, the **hooks/custom folders** are where we look for those plugins. In the next version of the system, that will allow overriding functionality as well. The current system **only allows** the specific hooks to be called.

The "stages" in the `circle.yml` file are specific to CCI, we did not come up with those and they have specific meaning to CCI (for instance, the test section will always run to completion, regardless of failure but other sections will bail on the first failure). Those sections may or may not have specific subsections (pre, post, test, override, etc,) depending on which section.

You can consult the now-defunct CCI 1.0 (yes, **CCI 1.0 is severely out of date**) to see what section/subsection is allowed and what it does, and then override as appropriate; where you should do that comes down to a few things:

1. Do you intend to swap-out functionality (i.e., use Jenkins or Travis instead of CCI)?
2. Do you want Canaveral to make updates to the core files when needed?
3. Does Canaveral actually do what you want it to do?
4. Is there an appropriate override for what you are trying to change?
5. Should this functionality be a service plugin?

The answer to those will determine what you will want to do.

A question we get a lot is **"Why do we need X ... when we can do the same thing from Y ..."** (e.g., "Why do we need Canaveral Engine when we can deploy from CCI?" or "Why use Github when we have Gerrit?"). The answer comes down to what *you* as the owner want to do and what you have available to you:

- If you *really* want to use PRs, then you need Github.
- If you *really* like CCI, then you also need Github.
- If on the other hand you really want *Jenkins*, then Github or Gerrit works.

When it comes to things within a segment, it is more about **resources and being a good citizen**. You *can* deploy from CCI, but you are **using up one of a limited number of containers (approx 60)** to do your deployment! If it takes a few minutes - no big deal; if it runs for hours - well, that's **one less build** others can do.

> In the case of resource usage or "abuse," **we will intervene to prevent fair usage**, but within that - we have no specific qualms with what you do in each segment.

Finally, even though systems in a segment have overlapping offerings (deploy from CCI/Jenkins or the Deploy Engine) the capabilities might differ significantly. For instance, the Deploy Engine is capable of pulling secrets, using jump boxes, etc. if you are in need of these it might make more sense to have Canaveral Deploy Engine do the work rather than handling it yourself in your CCI/Jenkins build.