Jenkins World 2017 Contributor Summit Notes

http://bit.ly/2x11CUZ

(or tinyurl.com/jw17contrib)

Context

- Agenda
  - 9:00 – 9:15 Welcome
  - 9:15 – 9:45 Introduction to participating in the Jenkins project
  - 9:45 – 10:45 The last year in review
  - 10:45 – 11:00 Break
  - 11:00 – 12:00 A vision for Jenkins
  - 12:00 – 13:00 Lunch
  - 13:00 – 13:30 Discuss unconference topics
  - 13:30 – 16:30 Unconference tracks/discussions
  - 16:30 – 17:00 Recap and wrap-up
- Let's jointly take notes
- WiFi: JenkinsWorld2017 / jenkinsbutler

Introduction to participating

Liam Newman: Intro to participation in Jenkins

- @bitwiseman
- *Anyone* who makes the project better is a contributor.
- http://jenkins.io/participate/
-

Eric Smalling: on organizing events

- I organize Dallas/Ft.Worth JAM
- Maxwell Arbuckle & Alyssa Tong own JAM Meetup.com account and will set up your JAM group.

- 50% of RSVP ex: 25 people turn-out are great show. Don't get discouraged feeling this is too small
- Having a regular venue is great. Try not to bounce around. That way, you ensure consistent showing of people
- Q&A has been great; I still learn new things — there are so many smart people
- We organize: 1hr talk, 30min Q&A, + 20min social
- How do you find speakers?
    - Eric: it's still tough for me
    - It shouldn't be just "Eric Smalling" show, it shouldn't be a vendor pitch show
    - Once you get people coming, you know who are "high caliber Jenkins people"
    - In the beginning, you have to do some talking yourself, and I also leaned on CloudBees to provide some speakers
    - Try to avoid "Vendor Pitch" talks
- What are the startup cost?
    - JAM program covers the meetup.com cost
    - I know some that basically "pass a hat" (i.e., take donations from attendees)
- What's the time commitment as an organizer?
    - Coming up with a topic: a couple of hours a week, make sure people are coming
- What can a project do better to serve JAM organizers?
    - Google folders with presentations used in other JAMs, perhaps with recording
    - Recycling talks is fine!
    - rtyler: it's already there! See https://github.com/jenkinsci/events-materials

Christopher Orr: on contributing code

- https://jenkins.io/doc/developer/extensions
- This project is open by default
    - Low barrier to entry
    - Transparent communication channels: IRC, project meeting
- Areas of contributions
    - Simple fix, by picking up JIRA label called "newbie-friendly"
    - Documentation
    - Translation
    - Plugins
        - Improve existing plugins - features & fixes
        - Adopt a plugin
            - Take place of people who have left maintaining plugins
        - Build a new plugin
        - See https://jenkins.io/doc/developer for various technical info
- Q&A
    - I find it difficult to get started on writing Jenkinsfile in my plugin repo
        - Basic instructions on this page: https://ci.jenkins.io/
        - echo 'buildPlugin()' > Jenkinsfile

        - More details under jenkins-infra/documentation
    - Bobby: I'd like to add to "build a new plugin" that you should ask us to see if we can find a similar/identical plugin
    - JamesN: the core-plugin version matrix, is this generation automated?
        - Daniel: they are auto-generated now but links don't exist yet
        - `stats.jenkins.io/pluginversions/(artifactId).html`

Last Year in Review: Daniel, Alyssa, Tyler

(Officer & team updates)

Security Officer: Daniel Beck

- *(Need to clean up slides for distribution.)*
- It was the best of times, it was the worst of times.
- https://jenkins.io/security/#reporting-vulnerabilities please
- https://jenkins.io/doc/developer/security/ for plugin devs (by no means complete)

Infrastructure Update: R. Tyler Croy

- Olivier is here! Working full-time on infrastructure.
- Azure migration is ongoing; lots of progress
    - ci.jenkins.io is running builds and tests for plugins
        - Windows support by default!
    - Lots of stuff running on Kubernetes on Azure
        - Plugins website, account app, other wee apps
- Infrastructure Enhancement Proposals: https://github.com/jenkins-infra/iep/
- Q&A
    - Oleg: Are we going to have core/plugin releases made from ci.jenkins.io?
        - We've concentrated on migrating the existing stuff
        - We've talked about this, but we should talk more about it; Oliver Gondža (release manager) has some ideas (but sadly he's not here at Jenkins World)

Events Update: Alyssa Tong

- CloudBees has Alyssa and Max to help make JAM organizers' job as easy as possible
- 2016: 45 groups, 8226 members
- 2017: 66 meetups, 17535 members
- Meetups list: https://meetup.com/pro/jenkins
- Project site: https://jenkins.io/projects/jam
- Existing presentations: https://github.com/jenkinsci/events-materials
- Mailing list: jenkinsci-jam@googlegroups.com
- https://docs.google.com/forms/d/1dGpwxpwoJDHR3fTlIcFXO8GZVpx5i_dWUlbi9LKolX4/viewform?edit_requested=true

-

10:45 Vision for Jenkins: Kohsuke, Tyler

Kohsuke: Let's put users first

    - "We're not adequately equipped to meet this challenge"
    - We do a lot of stuff well: building new integrations, catering to power users, building experiments as new plugins
    - Make Jenkins great for "the masses": new users, time-constrained users…
    - What can we actively do to find the frustrations that users have?
    - James: How about we disable executors on the master, since nobody should ever be doing that? This could help solve the problem of people not knowing about agents.
    - Connect to the "dark matter" users
        - People that don't use jenkins.io, mailing lists, don't come to JW, stick to the same Jenkins setup they had two years ago…
        - How can we make Jenkins better for them?
            - Communicating new features
            - Make Jenkins simpler for them; require less trial & error
    - We should:
        - Find and study these users
        - Expose less complexity in the core of Jenkins to users
        - Try to proactively inform users when we detect something is wrong
- Goal: make Jenkins easier
    - "I've only ever set up Jenkins twice: once when I created Hudson, and once for my home" — KK
    - Target: a new user with a Git repo should be able to use Jenkins and start auto-deploying their changes
        - In five minutes, with five clicks
        - Regardless of programming language or deployment environment
    - How about a "Launch in AWS" button, rather than having to download a .deb?
        - Tyler mentioned we have this for Azure on jenkins.io/download
            - https://aka.ms/jenkins-on-azure
    - Reduce babysitting required
        - We could auto-upgrade Jenkins, e.g. for security fixes
        - Set default executors to zero for the master
        - Ephemeral build agents out of the box
    - Why is setting up integrations so hard?
        - Why don't we have those nice {GitHub, Slack, whatever} OAuth pages that all other apps have?

        - Instead, users have to go into Jenkins, paste in various URLs and access tokens into our web UI…
    - In-app product notifications
        - e.g. Trello has a UI element at the top of the page telling you about new features
    - Could we extend the plugin ecosystem we have to Pipelines?

Tyler: Future of the Project and Community

- https://docs.google.com/presentation/d/1E3sUlRnfG-Dpmj-Lwrse56S0aUY3PBoGlenU5QwYCXg/edit?usp=sharing
- How should the project behave in order to continue to be relevant?
- It may be less about the code, and more about the community
- Encourage us to create the "Jenkins Enhancement Proposal" process
    - Similar to Python PEPs, and the IEPs we have in jenkins-infra
        - https://www.python.org/dev/peps/pep-0001/
        - https://github.com/jenkins-infra/iep
    - Formalised structure for architectural and technical improvements, or process within the project itself, or informational to describe existing design
    - Important: it's not "all talk," as there should be a reference implementation

Jenkins Enhancement Proposal (JEP)

- What's the boundary between JEP vs "just a bug fix / PR"
    - rtyler: I don't have a good answer, but think …
        - New API / deprecation of API → JEP
        - Bug fix → just a PR
- Editors understand the process of JEP and provide 1st level support to guide those who want to make changes
    - KK: and presumably to ensure consistency
- Decision making process is vague in Python
    - BDFL + its delegates
- This clarifies the process of driving consensus & making a big change, which helps larger participants to make bigger changes
    - Try and prevent contributors from saying "this is too complex; we're going to just maintain our changes in-house"

Governance

- Present day
    - Part of Software in the Public Interest
        - They hold the Jenkins trademark, handle donations
        - https://www.spi-inc.org/projects/jenkins/
    - We rely on a lot of patrons to "keep the lights on"
    - Current governance structure means that the Jenkins project can't sign contracts

        - Azure agreement happened with the individual board members
        - SPI was not willing to sign an NDA on the projects' behalf
    - Board elections are still outstanding, since 2011
- Idea: Jenkins Software Foundation!
    - Found a standalone legal entity
    - Various levels of membership
        - Premier (e.g. Microsoft, CloudBees, Red Hat, Google…)
        - General/individual (e.g. individual contributors)
        - Associate (e.g. OSUOSL, Apache Software Foundation…)
    - Governance board
    - Technical Steering Committee
        - New thing that we don't currently have
        - Responsible for driving JEPs, defining working groups
    - Marketing Committee
-
-
-

Unconference Topic Ideas

Slot 1: 13:30–14:20

Track #1 (this room): Automation of new Jenkins instances (Jenkins as a Service) in an organization

- NdL: discussing readable YAML format for defining Jenkins global config
- system-config-dsl from KK (using Groovy)
- Praqma/jenkins-coco-plugin
- `@Symbol` to map to YAML keywords, fallback to "natural" default (hudson.security.LDAPSecurityRealm -> "ldap")
- Basically static, except interpolating environment variables
- vs. use of Chef or Puppet etc.
- deny saves of overridden configuration
    - https://issues.jenkins-ci.org/browse/JENKINS-13190 could permit `Jenkins.CONFIGURE` to be denied to non-admins, akin to https://github.com/jenkinsci/workflow-multibranch-plugin/blob/c49261f827d032a637475071ba6742f0c40a8653/src/main/java/org/jenkinsci/plugins/workflow/multibranch/BranchJobProperty.java#L67-L78 for Pipeline-as-code
- GitHub orgs can handle the job config portion in some cases
- "bare Linux server with Docker daemon installed"—no need for agents per se
- credential handling: can create second-level secrets using whatever `master.key` was generated
    - Integrate w/ Vault or Docker secrets etc.

    - Possibility of never storing secrets in `$JENKINS_HOME` volume: http://javadoc.jenkins.io/jenkins/security/ConfidentialStore.html
- Java keystores, self-signed SSL certificates, etc.
- Export a live example `$JENKINS_HOME` to YAML, but without default values like https://github.com/jenkinsci/structs-plugin/blob/0194fee4c5d25406409141c5d1e2b2776a924758/plugin/src/main/java/org/jenkinsci/plugins/structs/describable/DescribableModel.java#L548-L568
- and/or generate schema à la https://jenkins.io/doc/pipeline/steps/ (static) or https://ci.jenkins.io/pipeline-syntax/ (dynamic)
- How does config get refreshed w/o loss of uptime? Or do you need to "reload from disk"?
- How do changes get tested before production? What is the validation test? Run some predefined jobs?
- What's preventing the existing plugins from KK and Praqma from becoming mainstream? Mostly lack of contributions
- Compatibility / upgrades: Java calls vs. `config.xml`; immutable jenkins config as code vs changing/mutating config (consider `readResolve` which keeps serial compat but not `DescribableModel` compat)

Track #2 (golden gate C3): Jenkins Software Foundation

- Chris Aniszczyk (@cra) from the CNCF (technically a subsidiary of the Linux Foundation) giving his experience on how this stuff works
- Does this let us have a code signing key named to the project?
- How are other projects doing?
    - Debian: they never had a corporate patron
- Technical Standards Committee
    - How this group forms need to be decided by the project
    - Some gets voted in by board
    - Some can get added by the people the board has voted in
    - There can be some seats selected by other means; such as representing users
- As somebody from investment bank, right now we cannot contribute to the project.
- Things like JAM, which currently — to some extent — rely on CloudBees marketing budget, should come under the Foundation umbrella and its budget.
- Apache vs Linux Foundation
    - Chris: I feel like Apache doesn't recognize that companies exist, that they are all individuals
    - Chris: Linux Foundation has full time professional staff employed, which allows us to provide higher touch services, each sub foundation has a different level of staff depending on sub foundation funding
- Exit strategy
    - Chris: you'd have it in the charter.

Slot 2: 14:30–15:20

New contributors' perspective (this room)

- Benjamin
    - I started with Jenkins at work, got into Pipeline multi-branch feature. Warning plugin didn't support multi-branch stuff, so I filed a bug and started digging in.
        - Guide existed how to adapt
        -
    - In my day job, I don't do Java, but I enjoy playing with other languages when I'm off clock
    - Started by reporting the bug
        - Maintainer response: No time, pointed to guide
        - Took a while to implement it, after several weeks opened PR
        - Response: Looks mostly OK, needs autotests. Maintainer was responsive
        - Experienced: Well maintained software project
    - Q: Did you know how to get started?
        - Yes, obvious. Just google it, found everything
    - Maven was a great burden, never used it before
        - Mainly around making changes in related projects and updating the dependencies for other dependant project for SNAPSHOT
        - Suggestion: Make it easier to use Maven, have some docs on that
    - Java wasn't that difficult, was able to copy stuff around
    - KK: What would it take to go to the next level on contributions?
        - Checks Jira for own issues
        - No plans to go deeper into core or similar
    - MW: Considered other kinds of contributions?
        - No, only code, scratched own itch
- Mark
    - Authenticating the first release properly was really hard; all the maven setting stuff
        - Trial and error of Maven artifact uploads with actual releases
            - mvn deploy a snapshot would work, but undocumented
            - ON: "verify" could be added to Maven HPI plugin
- Owen
    - GitLab plugin experience: Accepted every PR, breaking everything
    - Triage everything, respond to users, write docs, recruits new contributors to the plugin
    - Presentation at work on how to contribute other than writing code
    - MW: How do you evaluate code contributions without Java experience?
        - If it's a bug fix, reproduce and test, and no obvious mess, merge
        - Features may not fit the plugin and may be rejected
        - Other changes OM tries to get another maintainer involved, but they moved on from Gitlab

    - DB: have you used the Code reviewer team?
        - OM: No, didn't know
        - MW: I have but the team felt rather small
    - People have barriers to entry in their own minds
        - MW: Same problem, overcoming impostor syndrome
- People have perception that contributing to a project is hard and my change is not worthy
    - Asking people to volunteer empowers them
    - DB: does Adopt-a-plugin program help? Or do we need more personal reach-out to individuals?
        - OM: Adopt-a-plugin is great program as it encourages change
            - Invitation to take on a lot of responsibility
    - For core, slide's bug triage idea comes up but it tapers off
        - OM: more things like that which create "project is asking for help" will help getting contributors
    - MW: Bring people in who don't know to even ask
    - ??: Time between having an itch and having the courage to participate
    - OM: Other kinds of contributions, people might not know there's a need
        - E.g. Mic needed someone to review a Chinese translation for Blue Ocean
- Mike
    - https://github.com/mbarrien ?
    - Python dev by day
    - Employer uses Jenkins for 4-5 yrs, mostly classic UI
    - discovered Pipeline a while back, but many plugins weren't adapted (code cov, slack, …)
    - Converted some of these plugins to be PL-compatible
    - Cobertura plugin maintainer after long saga. Also a bit of EC2 plugin
    - Technical frustrations
        - Jenkins plugin API needs deeper knowledge of internal API
            - (Also EC2 plugin)
            - Just to do main functionality, need to know a lot
                -
            - For many of these things, had to read Javadoc/sources
        - Abstractions (class hierarchy) is difficult to understand
            - EC2Computer extending SomethingComputer which then extend Computer, …
            - Also around Pipeline stuff
                - Another person agreed
            - AbstractProject, Project, Job, AbstractItem
        - Deprecated methods all over the place
        - Could have benefitted from knowing about plugin usage stats
        - Transient fields / saving to disk
            - More specifically, around persistence behaviour in and around Pipeline.

        - Process level: Plugins were abandoned and therefore not adapted
            - Adopt-a-plugin existed, creating an account was a mess
                - GitHub, jenkins.io, Artifactory
                - Maintainers written in POM was not the actual maintainers
                - EMail to google group, wait to respond
                - Separate permissions for Artifactory and GitHub
        - Too many layers of abstractions, difficult to navigate the code, different GitHub repositories
            - When I need to look up a class I don't know which repo it exists
            - Lack of guide to GitHub repos
            - Lots of legacy build-up
        - I tend not to use IDEs to edit plugins — I use Sublime Text
            - DB: why?
                - All I'm trying to do is to scratch a little itch. I want to git-clone, modify a little thing here, build, and move on
        - Little experience with Java, knowing which repos to pull down
        - MW: Any recommendations how to share abstractions
            - I tried to search but didn't find it
        - MW: Any suggestions to simplify version updates?
            - Problems due to additional checks in newer parent POMs
            - Confusion between parent POM version and Jenkins dependency
        - KK: Did you feel there's a channel to look for help?
            - In EC2, asked on Google Group and gone to IRC
                - Asked some questions to IRC, didn't get answer, so after that I haven't gone there.
                - Didn't ask for help on the Google Group, just tried to become maintainer
    - Process problems (non-technical)
        - Didn't know how to ping people to get responses for maintainer request, but the process overall wasn't crazy
        - Account setup was split over several pages
        - Some comments on PRs were unclear (why make it transient?)
        - Presentation by orrc was useful, would have helped
            - How to submit PRs is clear, becoming a maintainer less so
    - KK: What the project should do going forward?
        - Cognitive overhead of deprecated methods should be removed
            - Deprecated APIs all the way down
        - Also contributed to other projects like salt stack, which is Python-based, concepts are easier to understand
        - We're too far on the side of backwards compatibility resulting in obsolete APIs
    - DB: Would a guide have helped with an IDE?

        - Starts out with minimal process (git pull) and just does it in the default editor
        - MW: Did the first year in emacs, needed to debug in an IDE, then noticed how useful this is
        - ??: Very new uses vim; knowing that an IDE makes things much easier would be helpful
- Contributors with non-Java background
    - KK: both Benjamin and Mike are in this camp
    - (There was another one in the audience who uses vi mainly)

How to automate core and plugin releases from ci.jenkins.io? (Golden Gate C3)

- ci.jenkins.io is untrusted, since pretty much anybody can run Pipelines on there
- trusted.ci.jenkins.io is a separate instance, which is not publicly available
    - This generates and signs the update site
- Automating core releases
    - Requires getting the signing key from Kohsuke
    - Signing key is currently pretty small / uses old ciphers?
- Automating plugin releases
    - More likely to be implemented at the moment
    - What are the aims of automating this?
        - Passing off computing power from your laptop
        - Security: want to know that the binary released correlates to a git commit
        - Prevent developers from messing up the release (whether intentional or not)
        - Ideological goal (nice to have): reproducible builds, where you can reproduce a .hpi SHA1 from source
    - How do we trigger a release?
        - Based on a git tag being created?
            - This requires trust / correct permissions on GitHub
        - A Jenkins job, where you have to be authenticated?
            - This jobs could let you choose the branch to release from, perhaps
            - CloudBees uses a similar system internally?
        - Jesse: continuously build plugin releases and push them to Artifactory
            - The Update Centre would publicise the latest “approved” version of these
            - To update the approved version, the maintainer would submit a PR updated the metadata
            - Question: how would this affect plugins depending on others?
    - How do we verify that the person triggering the release is permitted to do so?
        - Need to prevent people from altering their artifactId, committing it, then triggering a release — i.e. prevent overwriting plugins they’re not meant to
        - Certifying developers and plugins

- Could the release process be totally automated using github as the source of truth, for instance:
  - Push a tag
  - releases-or-something.jenkins.io detects it and triggers a release => that would give/tag your plugin the associated metadata telling the UC infra that this plugin is trusted, then:
    - For the end-users to benefit from this too, we would reuse this and add a nice green checkbox in the UC next to your plugin if was released automatically following that process.
    - (Tyler) To accommodate existing practice, the "old way" would still be allowed, but then as the plugins would look more trustable from users with those green things, developers would be positively pushed towards following the /blessed/ process to enter the list of "trusted plugins" (possibly to be seen as similar as the different widths you get in a browser depending on the level of certificate you bought/set up).
      - (Sam) => asks to make it not just a boolean, for future evolutions
- Could we build a simple web-app, where someone logs in with their Jenkins account?
  - From there, they could trigger a release, and the Jenkins stuff would happen on trusted infrastructure in the background
  - This avoids having to give plugin developers some sort of VPN access to a trusted CI instance (and all the admin/support headaches associated)
  - This (somewhat) builds on the idea discussed at previous events of having a Plugin Developers' Dashboard
- Open questions
  - How do we trust developers? Tie it to their GitHub account?
- Changes required
  - Update centre generation process
  - Plugin site display
  - Account app
    - Link (via OAuth) GitHub account to Jenkins account
    - Require that GitHub users have 2FA
    - Will be useful in general
  - Pipeline shared library
- If this were a JEP, who would be the people responsible?
  - Tyler for the infra changes, potentially
  - Oliver for the release side?

Slot 3: 15:30-16:20

Main room: Organizing / publishing reusable Pipeline libraries

- This should not be a KK talking show xD
- One of the important parts of Jenkins (from KK's perspective) is the ability to build up abstractions
    - Enabling somebody else to build on top of those abstractions
- There's no incentive to build abstractions (no Update Center equivalent for Pipeline Shared Libraries)
- Alvin Huang: Copying pipelines around build technical debt
    - Shared libraries makes it much easier
    - Devs don't care about Jenkins, want to write Ruby, Python, etc.
    - 15-20 modular steps in the shared library (checkout, create Jira, publish RPM, etc.)
    - Getting started: Use snippet generator, copy it into a groovy file, then iterate to generalize
    - Abstraction:
        - Create modular steps for specific tasks
        - Create a build wrapper that calls these steps
    - Identified common behavior between projects
    - I want to be able to present "steps" that I created in my shared libraries to show in snippet generator, etc., so that my users feel like they are the 1st class citizen
    - MW: on the same token, do you want to be able to hide steps that you don't want developers to use?
- Tyler: what if we create a blessed set of pipeline libraries?
    - << conversation about compatibilities that KK didn't grok >>
    - Standard library would create lower-entry contribution mechanism compared to Java+Maven+IDE world of today
    - KK: if we are to do this, what modules would we do first?
        - JIRA & GitHub interactions: updating those systems to leave records
            - JG: I don't think there's anything pipeline specific about it, so it should be ~~fully fledged plugin~~ no! an external tool
        - JG: for me, it's not immediately obvious what those modules are
        - JG: Fabric8 project created an interesting library
        - Tyler: Docker has another interesting library
    - JG: who's going to review, curate, and manage those libraries?
        - DB: this is no different from pipeline examples that we do today
    - Alvin: capturing output from shell step invocation and read it back
        - … until this became a part of the sh step
- There is a way to create plugins with just groovy code instead of all the java cruft: simple-build-for-pipeline-plugin
    - https://jenkins.io/blog/2016/04/21/dsl-plugins/
- What does it take to make libraries show up in Snippetizer & doc generator, etc?
    - KK/Sam: This seems doable
    - JG: This basically involves copying half the code of CloudBees template plugin

    - DB: I'm good with something low effort like just annotating parameters
    - Bobby: Blue Ocean must be doing something similar for the pipeline editor that should be applicable
- Alvin: I'm primarily asking for just being able to document my "step" I created in a library
- Alvin: and being able to use this in a declarative pipeline
    - KK: … and get all the benefits of validation, etc
- https://github.com/jenkinsci/jenkins-scripts/commits/master
- https://github.com/jenkinsci/pipeline-examples
- https://github.com/fabric8io/jenkins-pipeline-library
- Tyler: when a pipeline library is not in the jenkins repo, it's difficult for me to promote it in the official doc / evangelism effort because there's a limited trust to those
-
-

C3: Deep technical changes for tomorrow

- Sub-agenda:
    - Filesystem is not a database. XmlFile / TaskListener / Artifacts abstraction could allow us to get rid of Jenkins_home storage
    - How can we get rid of remoting? (Nicolas)
    - Obsolete core dependencies are more and more a concern. We know we will have to address this at some point. So what can we do?
    - Reworking Jenkins to modules/OSGi
    - Rework Jenkins to Microservice architecture
- Pluggable Storage
    - https://docs.google.com/document/d/1sE6BxkUpKCbII-lV-tC_rE97Qqi7jUF_7QJpX0IRZ2Q/edit
    - How to do it without breaking changes?
    - JN: Virtual Filesystem (?)
    - ON: More complex rework, to seems to be feasible to do it in a non-breaking way
    - SC: XStream? It is not parallelizable enough
- Breaking Changes
    - Two items:
        - API Deprecation
        - Classloader
    - ON: API Deprecation engine has been planned for Jenkins 2.0, never completed
    - Needs to be flagged in the plugin manager
    - JN, ON: Dry-run JEP by the API deprecation Engine proposal?
- Microservices
    - <all>: Spinnaker, Concourse CI: tools could build stuff on the top of Jenkins...
    - Would be useful to have more deployment capabilities integrated into Jenkins. ALM, not just a CD Pipeline

    - …
    - Takeaway: Jenkins is fine, still needs some more integration capacity fir ALM management
- API shading
    - HPI Plugin + Accmod?
    - OSGi (without dynamic !)
    - Takeaway: JEP it

Unassigned

- New contributors' perspective
    - As an old-timer, where can we improve to make the project more accessible to new people?
    - Way too much @Deprecated code. Policy for code removal ?
    - ++ ++ +Daniel
- Automation of new Jenkins instances (Jenkins as a Service) in an organization
    - Will this include just setting up one Jenkins cluster? (basic config mgmt)
    - + +++
- How to automate core and plugin releases from ci.jenkins.io?
    - Sam Gleske, Daniel Beck, R Tyler, Oleg, Jesse
    - ++++++
- Jenkins Software Foundation? (ChrisA from LF and whoever else is interested)
    - +1s: Oleg, samrocketman, batmat, rtyler, Daniel, ChrisA
    - ++
- Organizing / publishing reusable Pipeline libraries (Kohsuke's teaser idea)
    - +, +SamVanOort, +batmat + +Daniel
- Deep technical changes for tomorrow
    - Filesystem is not a database. XmlFile / TaskListener / Artifacts abstraction could allow us to get rid of Jenkins_home storage
        - +++++
    - How can we get rid of remoting? (Nicolas)
    - Obsolete core dependencies are more and more a concern. We know we will have to address this at some point. So what can we do?

(everything below here has few votes)

- What are we missing in infrastructure? Infra Wish List session
    - Accessing IEP/JEP documents from jenkins.io
    - +

- API commonality & workflow steps (order & naming of parameters) - should we have a “API / step guide”
    - More coding convention to make code more accessible
        - +
    - Do we need to have some code review for new plugins? (a sucky popular plugin can make Jenkins look really bad)
- Oleg: Jenkins “Core Team” - establishing better review/release process for Jenkins core and detached plugins
    - TL;DR: Have a team, which shares responsibility for the core things + has a more formal process for reviews and disagreement resolution. @code-reviewers works in general, but we need to push it beyond just code reviews
    - Oleg: it’s well aligned with the JEP proposal
    - +
- Filesystem is not a database. XmlFile / TaskListener / Artifacts abstraction could allow us to get rid of Jenkins_home storage
    - +++++
    - Notes from the previous year: https://docs.google.com/document/d/1sE6BxkUpKCbII-lV-tC_rE97Qqi7jUF_7QJpX0lRZ2Q/edit
- How can we get rid of remoting? (Nicolas)
    - +
- Obsolete core dependencies are more and more a concern. We know we will have to address this at some point. So what can we do?
    - New libraries - how to not expose them
    - Deprecating and replacing libraries
    - +Sam Van Oort
-
- Security 101 for plugin maintainers (Daniel)
    - +++
- Deployment Management capabilities/patterns built in or plugin (targeting common cloud platforms: AWS, k8s, etc.)
- Extension for plugin’s statistics (allow a plugin to push custom stats through std infra) -- batmat (JENKINS-32485)
    - +1s: batmat +
- GSoC 2018 & 2017 retrospective. What could we do the things better? (Oleg)
    - +1s: oleg

Retrospective for Contributor Summit 2018

- Vishal: have more first time contributor speak up & share their experience
    - +

- Vishal: Regular channel to hear from existing senior contributors would be useful
    - +
- Benjamin: we should introduce each other. Could be even just saying the name
    - +
- Oleg: 1 day is not enough, consider 2-day summits instead
    - +
- Jesse: call for unconference topics in advance, preferably using Trello
    - +
- Definition of best practices
- Need some concrete actionable results at the end

7:30

“I wanted to let you know that we are providing limited taxis to the Autodesk Party, circling between the Marriott Marquis and the Autodesk Gallery. The taxis will be branded "Jenkins World 2017" and will run from 7pm to 8pm only.” — Alyssa

Autodesk Gallery
1 Market Street
#200
San Francisco, CA 94105
View Map