Continuous Integration Tools for Android

Continuous integration is a coding philosophy and set of practices that drive development teams to implement small changes and check in code to version control repositories frequently. Because most modern applications require developing code in different platforms and tools, the team needs a mechanism to integrate and validate its changes. The technical goal of CI is to establish a consistent and automated way to build, package, and test applications. With consistency in the integration process in place, teams are more likely to commit code changes more frequently which leads to better collaboration and software quality.

Advantage and Disadvantages of Continuous Integration Tools:

Jenkins:

Jenkins is an open-source continuous integration server written in Java. It is by far the most widely used tool for managing continuous integration builds and delivery pipelines. It helps developers in building and testing software continuously. Basically, Jenkins integrates development life-cycle processes of all kinds, including build, document, test, package, stage, deploy, static analysis and much more.

With Jenkins you can configure alerts in several ways, for example, you can receive email notification, pop-ups, etc. and actually automate it. By implementing the right configuration for you, you get almost immediate feedback. You will always know if the build broke. You will get to know what the reason for job fail was and you can also get to know how you can revert it back.

Advantages of Jenkins:

- It is open source and it is user-friendly, easy to install and does not require additional installations or components.
- It is free of cost.
- Easily Configurable. Jenkins can be easily modified and extended. It deploys code instantly, generates test reports. Jenkins can be configured according to the requirements for continuous integrations and continuous delivery.
- Platform Independent. Jenkins is available for all platforms and different operating systems, whether OS X, Windows or Linux.
- Rich Plugin ecosystem. The extensive pool of plugins makes Jenkins flexible and allows building, deploying and automating across various platforms.
- Easy support. Because it is open source and widely used, there is no shortage of support from large online communities of agile teams.
- Developers write the tests to detect the errors of their code as soon as possible. So the developers don't waste time on large-scale error-ridden integrations.
- Issues are detected and resolved almost right away which keeps the software in a state where it can be released at any time safely.
- Most of the integration work is automated. Hence fewer integration issues. This saves both time and money over the lifespan of a project.

Disadvantages of Jenkins:

- The costs of hosting the server (which isn't free) that Jenkins runs on cannot be predicted easily. Hence the cost factor, even if Jenkins itself is free, remains unpredictable.
- Jenkins' management is generally done by a single user and that leads to tracking and accountability problems with the pushed code.

- Jenkins doesn’t provide any analytics (there are plugins but they are not enough) on the end-to-end deployment cycle. This again goes back to the lack of overall tracking that contributes to the lack of analytics as well.

TeamCity:

TeamCity is a Java-based build management and continuous integration server from JetBrains. It is a powerful continuous integration tool.

It is known for its incredibly simple setup and beautiful user interface. It has a robust set of features out of the box and a growing plugin ecosystem.

Advantages of TeamCity:

- TeamCity provides a great integration with git, especially Bitbucket.
- When a new code release (build) fails TeamCity has a great tool for investigation and troubleshooting.
- TeamCity provides a user-friendly interface. While some technical knowledge is required to use TeamCity, the design helps simply things.

- Easy to set up. The UI is pretty easy to navigate and use. You can have your project up and running in minutes.
- Good integration with various build frameworks/methodologies. You can run standard Maven, Ant or Gradle builds with virtually no customization.
- Price. It's free for limited use, so you don't need to pay until you ramp up and are using it a lot.

Disadvantages of TeamCity:

- Upgrading TeamCity is a long and manual process.
- Java skills are needed to fully utilize TeamCity, although they are not necessary for basic or medium-level use.
- Log formatting could be a little clearer, though that is true for almost all build systems.

Bamboo:

Bamboo is a continuous integration (CI) server that can be used to automate the release management for a software application, creating a continuous delivery pipeline.

Advantages of Bamboo:

- Build Automation
- Continuous Integration
- Configuration Management
- Continuous Deployment
- Integrations with the rest of the Atlassian suite - specifically bitbucket.
- Fast, automated build workflow...very hands-off for us.

Disadvantages of Bamboo:

- They could improve their pricing model for smaller teams
- They can focus on mobile applications
- The build plans seem to fail quite frequently. This occurs for no apparent reason. If it fails once, we will attempt to deploy again. Usually, after 2-3 times it will work.

Circle CI:

Circle CI is the next cloud-based CI/CD tool in our list. It enables you to automate your build, test and delivery process. The mission of Circle CI is to give people everywhere the power to build and deliver software at the speed of imagination. To make that happen, Circle CI provides quick setup, integration with a range of tools, support for all test frameworks and the ability to easily configure the entire setup. Ultimately, it allows users to ship healthy code faster, safely and at scale.

Advantages of Circle CI:

- Automated builds! This is really why you get CircleCI, to automate the build process. This makes building your application far more reliable and repeatable. It can also run tests and verify your application is working as expected.
- Simple. Unlike Jenkins, Teamcity, or other platforms, CircleCI doesn't need a lot of setup. It's completely hosted, so there's no infrastructure to set up. The config file does take a bit to understand, but if you follow their example and start with something small and add to it, you can get it up and going quicker than it first looks.
- Scales easily. Again, since it's all cloud-based, you don't have to manage or scale infrastructure. Simply subscribe to the number of containers you want, and scaling up just means buying more containers.
- Parallel testing - run your unit tests in parallel containers shared across all branches Price - You pay by concurrent containers. We're currently using 10 to run unit test across dozens of branches for a completely reasonable price. No caches.

.

Disadvantages of Circle CI:

- No static IPs. This could cause problems if you want to enable only CircleCI to access your environment. Much of the limitations for us were around this issue, since we're in such a regulated industry.
- The search feature needs improvement. If you're doing a lot of builds, the history can go on for pages. We didn't find it suited our needs for audits/reports as you can't search by a particular developer who triggered a build, filter for only successful builds, etc.
- Slightly limited customization, something like Jenkins is more flexible. CircleCI used to have a very defined build process, but now with the introduction of workflows, it's gotten a lot better. I think they hit the right balance between simplicity and flexibility though. If you need a lot of integrations or other things that they don't offer, Jenkins is probably better. CircleCI isn't intended for complex applications, it's really about keeping it simple so you can focus on code development.

GoCD:

GoCD is an Open source Continuous Integration server. It is used to model and visualize complex workflows with ease. This CI tool allows continuous delivery and provides an intuitive interface for building CD pipelines.

Supports parallel and sequential execution. Dependencies can be easily configured. Visualize end to end workflow in realtime with Value Stream Map and deploy to production securely.

Advantages of GoCD:

- Easy Setup for deployment pipeline
- Environment Variables for each step
- Supports both Windows and Linux agent
- Highly customizable

Disadvantages of GoCD:

- UI can be improved.
- Location for settings can be re-arranged.
- API for settings up pipeline.

BuildBot:

Buildbot is a software development CI which automates the compile/test cycle. It is widely used for many software projects to validate code changes. It provides distributed, parallel execution of jobs across different platforms.

Buildbot comes with a ready to use docker container available at buildbot/buildbot-master Buildbot is easy to set up, but very extensible and customizable. It supports arbitrary build processes, and is not limited to common build processes for particular languages (e.g., autotools or ant).

Advantages of BuildBot:

- It can parse the output of trial and provide summary reports that enumerate the problems with a build.

- It can build specified branches, and along with force-builds.py it provides a reasonably nice interface to both submitting and viewing the results on a particular branch.
- It's very easy for third parties to do the initial setup of a build slave, allowing people to contribute hardware with relatively little friction.
- Builders can be segregated into "supported" and "unsupported", allowing us to have at-a-glance views of both the builders that should be green and the ones we hope will be green eventually.

Disadvantages of BuilBot:

- The web status is rendered from a significant quantity of data stored in pickles on disk. The entire rendering process is expensive and no results are ever cached.
- Access to files is often denied (apparently spuriously) on Windows, causing source checkouts to fail, test runs to fail, individual test methods to fail, etc. The consequences range in severity from a failed build which is never retried to a build with a failed test which is irrelevant noise.
- There are no reports, or reporting functionality, such as tracking or graphing of statistics, such as "number of failed builds per week" or "average test-run time". The difficulty of accessing the build history as structured data (due to the aforementioned massive-pile-of-pickles problem) makes it difficult to implement any custom reporting.