Articles
- DevOps Principles
- DevOps Frameworks
- DevOps Tools
  - Overview
  - Considerations for your DevOps toolchain
  - DevOps Monitoring
  - DevOps Pipeline
  - DevSecOps Tools
  - Test Automation

Tutorials
- Automation
- Testing
- Security
- Observability

DevOps Tools

Choose tools for each phase of the DevOps lifecycle.

DevOps is the next evolution of agile methodologies. A cultural shift that brings development and operations teams together. DevOps is a practice that involves a cultural change, new management principles, and technology tools that help to implement best practices.

When it comes to a DevOps toolchain, organizations should look for tools that improve collaboration, reduce context-switching, introduce automation, and leverage observability and monitoring to ship better software, faster.

There are two primary approaches to a DevOps toolchain: an all-in-one or open toolchain. An all-in-one DevOps solution provides a complete solution that usually doesn't integrate with other third-party tools. An open toolchain can be customized for a team's needs with different tools. Atlassian believes an open toolchain is the best approach since it can be customized with best-of-breed tools to the unique needs of an organization. Using this approach often leads to increased time efficiency and reduces time to market.

Read more about DevOps toolchains.

Regardless of the type of DevOps toolchain an organization uses, a DevOps process needs to use the right tools to address the key phases of the DevOps lifecycle:

- Plan
- Build
- Continuous integration and deployment
- Monitor
- Operate
- Continuous feedback

With an open DevOps toolchain, the selected tools touch multiple phases of the DevOps lifecycle. The following sections showcase some of the most popular tools for DevOps, but given the nature of the market, this list changes frequently. Providers add new capabilities that enable them to span more phases of the DevOps lifecycle, new integrations are announced each quarter, and in some cases, providers consolidate their offerings to focus on a specific problem for their users.

Plan

Jira Software, Confluence, Slack

Taking a page out of the agile handbook, we recommend tools that allow development and operations teams to break work down into smaller, manageable chunks for quicker deployments. This allows you to learn from users sooner and helps with optimizing a product based on the feedback. Look for tools that provide sprint planning, issue tracking, and allow collaboration, such as Jira.

Another great practice is continuously gathering user feedback, organizing it into actionable inputs, and prioritizing those actions for your development teams. Look for tools that encourage "asynchronous brainstorming" (if you will). It's important that everyone can share and comment on anything: ideas, strategies, goals, requirements, roadmaps and documentation.

And don't forget about integrations and feature flags. Wherever you decide to scope your feature or project, it should be converted into user stories in your development backlog. Feature flags are if-statements in the code base that enable teams to turn features on and off.

For more on this phase, check out this post from Atlassian product managers about backlog grooming and prioritization.

Build

Continuous integration and delivery

Continuous integration: Jenkins, AWS CodePipeline, Bitbucket Pipelines, CircleCI, Slack, Snyk, Sonarsource

Continuous integration is the practice of checking in code to a shared repository several times a day, and testing it each time. That way, you automatically detect problems early, fix them when they're easiest to fix, and roll out new features to your users as early as possible.

Code review by pull-requests requires branching and is all the rage. The DevOps North Star is a workflow that results in fewer and faster branches and maintains testing rigor without sacrificing development speed.

Look for tools that automatically apply your tests to development branches, and give you the option to push to main when branch builds are successful. Along with that, you get continuous feedback through real-time chat alerts from your team with a simple integration.

See how Bitbucket Pipelines helps you automate your code from test to production.

Test: Mabl, Saucelabs, Xray, Zephyr

Testing tools span many needs and capabilities, including exploratory testing, test management, and orchestration. However, for the DevOps toolchain, automation is an essential function. Automated testing pays off over time by speeding up your development and testing cycles in the long run. And in a DevOps environment, it's important for another reason: awareness.

Test automation can increase software quality and reduce risk by doing it early and often. Development teams can execute automated tests repeatedly, covering several areas such as UI testing, security scanning, or load testing. They also yield reports and trend graphs that help identify risky areas.

Risk is a fact of life in software development, but you can't mitigate what you can't anticipate. Do your operations team a favor and let them peek under the hood with you. Look for tools that support wallboards, and let everyone involved in the project comment on specific build or deployment results. Extra points for tools that make it easy to get Operations involved in blitz testing and exploratory testing.

Deployment dashboards: Jira Software

One of the most stressful parts of shipping software is getting all the change, test, and deployment information for an upcoming release into one place. The last thing anyone needs before a release is a long meeting to report on status. This is where release dashboards come in.

Look for tools with a single dashboard integrated with your code repository and deployment tools. Find something that gives you full visibility on branches, builds, pull requests, and deployment warnings in one place.

Automated deployment: Bitbucket Pipelines, CodeDeploy

There's no magic recipe for automated deployment that will work for every application and IT environment. But converting operations' runbook into a cmd-executable script using Ruby or bash is a common way to start. Good engineering practices are vital. Use variables to factor out host names – maintaining unique scripts or code for each environment is no fun (and misses half the point anyway). Create utility methods or scripts to avoid duplicated code. And peer review your scripts to sanity-check them.

Try automating deployments to your lowest-level environment first, where you'll be using that automation most frequently, then replicate that all the way up to production. If nothing else, this exercise highlights the differences between your environments and generates a list of tasks for standardizing them. As a bonus, standardizing deploys through automation reduces "server drift" within and between environments.

Operate

Application and server performance monitoring: Appdynamics, DataDog, DynaTrace, Slack, HostedGraphite, Nagios, New Relic, Opsgenie, Pingdom, Splunk, SumoLogic

There are two types of monitoring that should be automated: server monitoring and application performance monitoring.

Manually "topping" a box or hitting your API with a test is fine for spot-checking. But to understand trends and the overall health of your application (and environments), you need software that is listening and recording data 24/7. Ongoing observability is a key capability for successful DevOps teams.

Look for tools that integrate with your group chat client so alerts go straight to your team's room, or a dedicated room for incidents.

Incident, change and problem tracking: Jira Service Management, Jira Software, Opsgenie, StatusPage

The keys to unlocking collaboration between DevOps teams is making sure they're viewing the same work. What happens when incidents are reported? Are they linked and traceable to software problems? When changes are made, are they linked to releases?

Nothing blocks Dev's collaboration with Ops more than having incidents and software development projects tracked in different systems. Look for tools that keep incidents, changes, problems, and software projects on one platform so you can identify and fix problems faster.

Continuous Feedback

GetFeedback, Slack, Jira Service Management, Pendo

Customers are already telling you whether you've built the right thing – you just have to listen. Continuous feedback includes both the culture and processes to collect feedback regularly, and tools to drive insights from the feedback. Continuous feedback practices include collecting and reviewing NPS data, churn surveys, bug reports, support tickets, and even tweets. In a DevOps culture, everyone on the product team has access to user comments because they help guide everything from release planning to exploratory testing sessions.

Look for applications that integrate your chat tool with your favorite survey platform for NPS-style feedback. Twitter and/or Facebook can also be integrated with chat for real-time feedback. For deeper looks at the feedback coming in from social media, it's worth investing in a social media management platform that can pull reports using historical data.

Analyzing and incorporating feedback may feel like it slows the pace of development in the short term, but it's more efficient in the long run than releasing new features that nobody wants.

In conclusion...

At Atlassian, we believe in the importance of having a DevOps toolchain that integrates with the tools development and operations teams love to use. That's why we built our DevOps platform to integrate with more than 171 leading third-party vendors, empowering you to make the best decisions across the tools you use. Because DevOps shouldn't be bought from a single vendor, but built.

To get started, try Atlassian's DevOps solution for free.

NEXT ARTICLE

Considerations for your DevOps toolchain →

Recommended reading

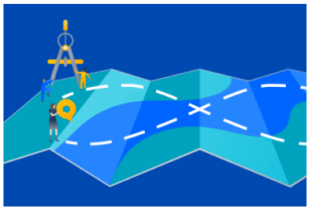
Bookmark these resources to learn about types of DevOps teams, or for ongoing updates about DevOps at Atlassian.

DevOps community

Learn more →

Simulation workshop

Learn more →



Get started for free

Learn more →

Sign up for our DevOps newsletter

Email address

Sign up