HOW TO SUPERCHARGE MAINFRAME DEVOPS WITH GIT + BMC AMI DEVX CODE PIPELINE

More and more mainframe organizations are either moving to Git or talking about moving to Git. Traditionally, the mainframe has existed as a separate platform, with its own set of tooling and processes, walled off from other platforms in the company like cloud, mobile, or web. As more mainframe teams are introduced to DevOps, though, these walls are starting to come down.

Many organizations use distributed tooling like Jenkins or GitHub Actions to orchestrate their continuous integration/continuous delivery (CI/CD) pipelines. Many also use tools like SonarQube to scan their source code and Veracode to enforce security. It is only natural that organizations would start to look to Git, the de facto standard for version control and source code management (SCM), to house their source. But this alone will not bring results—you need something to handle the remainder of the DevOps processes, such as build and deploy. To take your DevOps to the next level, you need to pair Git with a world-class build and deployment system like BMC AMI DevX Code Pipeline to supercharge your DevOps.

Why Git, why now?

The choice of Git is an easy one. It is the dominant SCM system in the industry and there are many reasons to adopt it.

- It was designed with the developer experience in mind. Whether you are using native Git or one of the popular remotes available like GitLab, GitHub or Bitbucket, Git was made to provide the best possible user experience and make day-to-day development easy. Git provides many built-in features to enhance the developer experience, like comparing, merging, and making it

easy to approve changes.

- Git supports multiple languages and multiple methodologies for working. Just as it supports distributed languages like Java, C, Node or Python, it can support Cobol, JCL, Rexx, PL1, Assembler or any other source language necessary for mainframe applications.
- It offers branching support to enable parallel development amongst teams and individuals. Git allows a user to create an isolated environment for development or R&D and then merge that code back into the main branch or trunk.
- Git allows organizations to consolidate their software delivery lifecycle (SDLC) processes across the organization, regardless of platforms. The mainframe will no longer be a siloed platform if it uses the same tool and process as the rest of the organization.
- It enables easier on-boarding of new developers and empowers less-experienced engineers on the mainframe platform. College grads and younger engineers already have experience, having been trained on Git systems from day one. They come out of school already having a GitHub account or equivalent and have been using Git in their studies.
- Git Platforms are open and easy to integrate into almost any system. Due to its prolific use, almost any DevOps tools or environment already supports Git and Git systems support most platforms and tools.

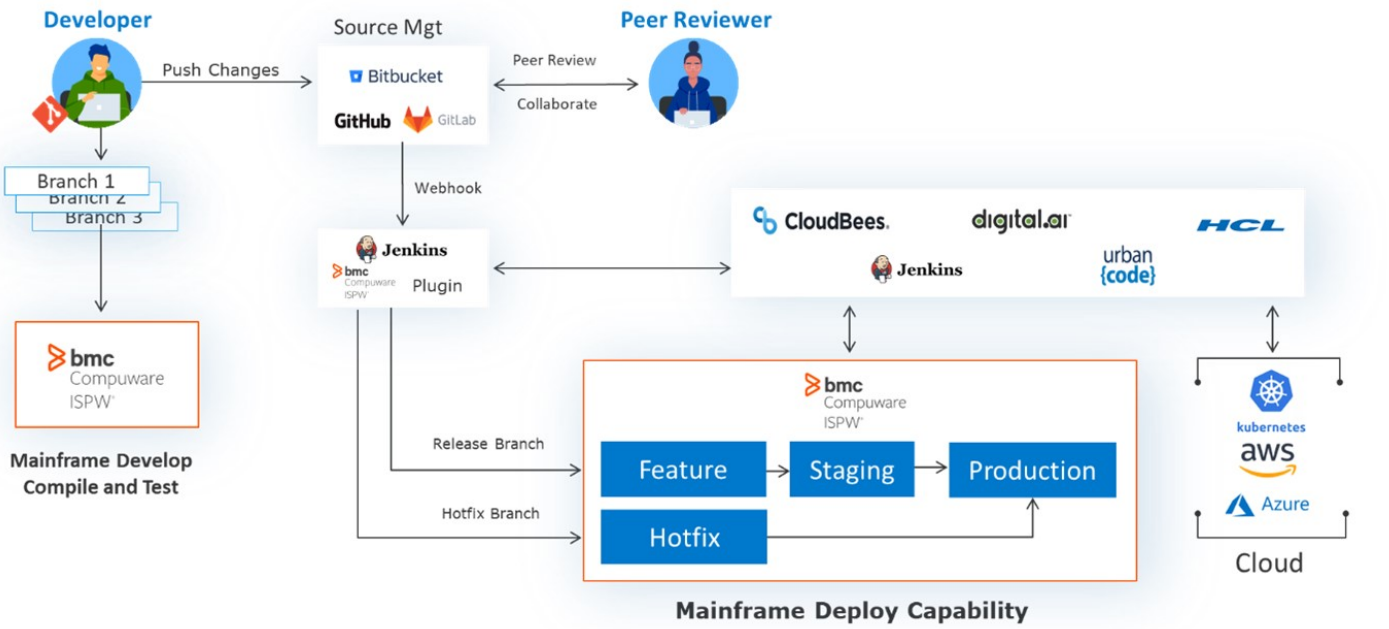
How does Git work with ISPW?

So, now we understand Git but how does it work with BMC AMI DevX Code Pipeline to create a next-level DevOps system? You may be thinking, "Doesn't ISPW handle source code itself?" Yes, ISPW does handle source code in addition to providing speed, agility, quality, and velocity in the way it handles that source. But, as detailed above, there are a lot of reasons an organization would want to bring Git into the picture to complement ISPW, allowing Git to do what it does best on top of ISPW's world-class build and deploy capabilities on the mainframe.

To use Git on the mainframe is no different than using Git on any other platform or with any other codebase. A developer would branch code into a local environment and start making changes. As part of that process, they would call ISPW to handle building and executing any unit tests in their local environment. Once complete and satisfied with their changes, the developer would issue a pull request to merge the changes into the main branch. At that point, the changes would be reviewed and, if complete, merged back into the main branch. This is no different than a distributed developer working in Java.

This is not the only interaction ISPW would have with Git though. Now that changes are complete and merged into the main branch, they must be deployed. Thanks to ISPW's open-borders approach, all the functions necessary to promote and deploy your change are exposed via API so now ISPW can be used to deploy your changes, as well. To make this even easier, a Jenkins plugin has been developed to handle this task which can easily be triggered from your Git system. If you do not have Jenkins, the same integrations have been built for other popular orchestration tools like GitHub Actions and Azure DevOps to name a few. Along with an API for anything, our expert services teams can help you create a deployment flow for your pipelines.

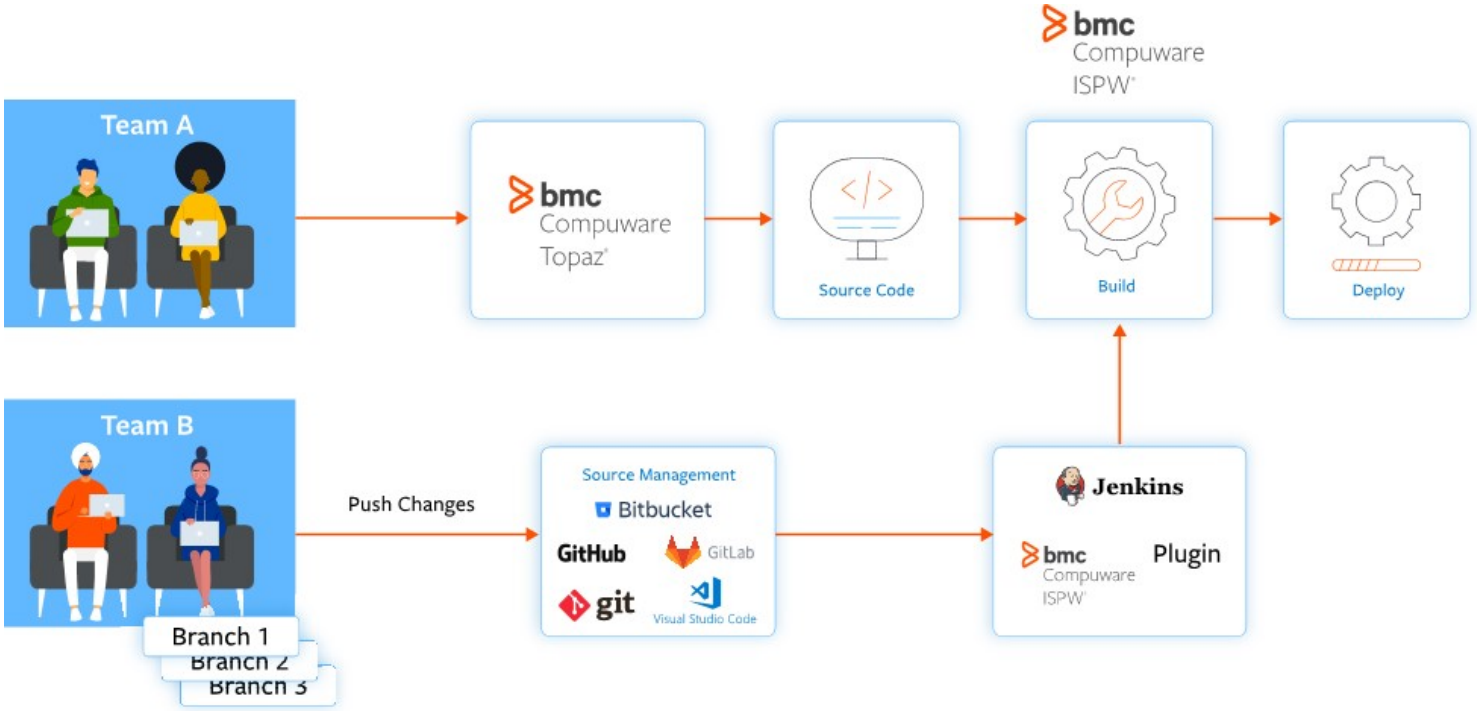
With your Mainframe plugged into the same process and CI/CD tooling you use for the rest of the platforms in your organization, you can then coordinate deployment across cross-functional teams so changes that go onto the mainframe can also be deployed at the same time as changes on cloud or any other distributed systems.



Does everyone HAVE to move to Git?

The answer is a simple, “no.” Like anything, this is not a one-size-fits-all solution. Git has many benefits and can be used in any scenario, but it might not fit every team or application. With BMC AMI DevX Code Pipeline, you can take a hybrid approach to development within the organization, allowing teams to decide if moving to Git is right for them. Let’s look at an example with 2 teams, team A and team B. Team A supports an application that is in maintenance mode, with no real new development, and they only deploy patches and updates 1-2 times per year. Team B supports a very active app that is deploying monthly, weekly, or maybe even daily. For team A, transitioning to Git does not make a lot of sense and they would get limited benefit out of it. So, for team A, using the traditional mainframe development method would be fine.

Team B, however, wants to move to Git and would reap many benefits from the capabilities Git brings to the table. Typically, either team A or team B would have to conform to the other teams wishes—but not with ISPW. ISPW allows for a hybrid approach to development where one team can use Git and the other team can use ISPW for all of their development needs. This allows for flexibility within the organization and provides choice so that each team can make the best decision for themselves.



Where should I start?

After reading all of this you might be asking yourself, "Where do I get started? Do I just put some code on Git and start working?" You can but that might not be the best place to start. Like anything, a phased approach is preferred to reduce the risks inherent in a major transformation. With a proven track record of success, BMC has some steps to follow to help you migrate successfully to using Git and supercharge your mainframe DevOps.

- **Step 1:** Migrate off your legacy mainframe SCM system and onto BMC AMI DevX Code Pipeline. Whether you have another SCM solution or a home-grown tool, getting off that system and onto a modern tool built for agility, like ISPW, is a key first step to setting yourself up for success. If you already have ISPW then you are 1/3 of the way there already. If not, BMC professional services can help migration from your old system so you can immediately start reaping the benefits.
- **Step 2:** Automate all your processes and get your pipelines built out. Integrate your existing DevOps tooling with BMC AMI DevX Code Pipeline and start automating all you can. Get your teams used to automation and DevOps and start making your process as efficient as you can.
- **Step 3:** Assess all your teams and begin migrating them to Git as needed. You have already set up your pipelines and automated your processes, now it is time to assess your teams and move those that want to migrate onto your Git offering and plug into the pipelines defined for ISPW. Once again, BMC professional services has scripting that can help you move your source code, history, and meta data to Git.

Using Git and BMC AMI DevX Code Pipeline together enables organizations to manage mainframe workloads with the speed and efficiency that only DevOps can bring. Doing so ensures that code is maintained in a place that encourages parallel development and is immediately comprehensible, while also allowing seamless and accurate building, testing and deployment of code.

To explore these concepts in greater detail, download our eBook, *Git for the Mainframe*.