DEVOPS FOR THE ENTERPRISE: IT TEAM CONVERGENCE AND IMPLEMENTING ON AWS

Patrick McClory July 28, 2016 DevOps 7,762 Views

As we've discussed throughout this blog series, the process of organizing teams, application artifacts, and technology processes under a DevOps strategy must encompass all teams contributing technical work to the macro-level system and should include the designing and developing of each team's software, tests, and automation processes. It also requires the use of a source control management system to retain the history of updates over the course of time. While this has been common practice for most software development teams for years, this might be a new practice for operations, test, and release management teams.

By placing all assets related to an application in a single source control repository, teams can easily recreate the state of the application for issue resolution debug and testing. Paired with a consistent, well-architected database snapshot strategy, even the database (with its data intact) can be restored to provide a closely representative copy of the production environment that can be used to fully investigate the root cause of any issue from deployment process failure through to end-user bugs introduced by a given version of code.

The paradigm of cloud computing enables a number of novel strategies in managing resources that were previously cost prohibitive, difficult to orchestrate or manage, or simply had no feasible technical implementation. The seemingly endless storage capacity in the cloud allows for strategies that leverage both short and long-term usage of massive amounts of disk capacity and server resources that, in a standard data center, would require ownership of the entirety of the capacity needed—a number that is difficult to plan for and inherently self-limiting. Lifting these hard capacity limits by working within the AWS platform and gaining the ability to rapidly restore and debug issues with actual data quickly and efficiently is only possible by leveraging automation, api-based access to resources, and infrastructure code that is versioned alongside the application code.

Consider the example of database backup management within the software release cycle. The backup of an RDBMS system before applying updates in a production system is good practice to ensure that there is a viable rollback plan in the event that an update to the database or application fails. In legacy architectures, the amount of time required to generate a standard backup, even for modest databases, adds meaningful time to the release process along with a disruption to the performance of the system for the entirety of the time the backup is executing. In the event that the release does fail, restoring that same backup might also take significant time, wherein the system will be fully unavailable to users.

In contrast, using EBS volume-level snapshots for the data and log volumes for the RDBMS, the amount of time required to create the backup is only as long as it takes for the system to pause writes and flush the write buffers. This, in contrast to RDBMS-based backup processes, consumes far less time by orders of magnitude and introduces only a slight impact to the systems' performance for a much shorter period of time. In terms of the ease of use in restoring the snapshot, the amount of time required to reconstitute snapshots is significantly less than that of having to restore or rebuild an RDBMS server from a backup file, often only requiring a reattachment of the restored volumes and a restart of the database server itself. Additionally, leveraging more managed services such as Amazon RDS can simplify the operational management of the database servers themselves and provide a far simpler interface for managing database snapshots in one place.

Knowing how snapshots for volumes operate along with the nominal additional operational cost compared to other backup methodologies, the concept of versioning production database snapshots is far less of a daunting and costly endeavor than traditional backup processes.

The strength of a DevOps strategy and an enterprise's ability to fully leverage the benefits of realigning its IT organization is heavily dependent on the organization's cloud strategy. Utilizing features such as volume snapshots and removing the hard-limit ceiling on storage and server provisioning is only possible by breaking out of the physical data center into the public cloud. The more features, configuration options, and architectural strategies and implementation choices that are available to DevOps teams, the more creative they can be with their technical solutions while still being cost-conscious. AWS provides most capacity with the richest Infrastructure as a Service portfolio available on the market today and has been noted as the clear leader in Gartner's Magic Quadrant.

Over the course of this blog series, we have talked through the evolving nature of DevOps theory and identified best practices for both team and project structure and alignment. In the end, we can clearly see that combining existing, familiar toolsets, and ownership-focused and technology-aligned teams with the flexibility and on-demand nature of the Cloud nets the perfect catalyst for enabling DevOps in the enterprise.