



The government seeks individual input; attendees/participants may provide individual advice only.

Middleware and Grid Interagency Coordination (MAGIC) Meeting Minutes

July 11, 2018, 12-2 pm
NCO, 490 L'Enfant Plaza, Ste. 8001
Washington, D.C. 20024

Participants (*In-Person Participants)

Bob Bonneau (DoD/OSD)	David Martin (ANL)
Richard Carlson (DOE/SC)*	Thomas Morton (DoD/OSD)*
Kaushik De (UTA)	Rajiv Ramnath (NSF)
Jill Gemmill (Clemson)	Don Riley (UMD)
Dan Gunter (LBNL)	Sonia Sachs (DOE/SC)
Marshall Lamb (IBM)	Alan Sill (TTU)
Joyce Lee (NCO) *	Kevin Thompson (NSF)
Miron Livny (UW-Madison)	Andrew Younge (Sandia)
Nitin Madhok (Clemson)	

Action Items:

- Follow up with speaker suggestions and continue fleshing out tasking.
- June 2018 minutes were approved and will be posted to the website.

Proceedings

This meeting was chaired by Richard Carlson (DOE/SC) and Rajiv Ramnath (NSF).

Speaker Series: Introduction to DevOps

- *Enterprise to Cloud: A DevOps Journey* – Marshall Lamb, Distinguished Engineer and CTO, Watson Supply Chain, Watson Customer Engagement Division, IBM Corporation
- *DevOps* - Nitin Madhok, Systems Architect, Clemson University
- *Leveraging Containerization for DevOps with Sandia's HPC Workloads* - Andrew J. Younge PhD, Senior Member of Technical Staff, Sandia National Laboratories
- *Innovative DevOps*: Tom Morton, Senior Analyst for Cloud Strategy and Policy, Office of the DoD CIO

Speaker Presentations

Enterprise to Cloud: A DevOps Journey - Marshall Lamb

Background

In 2013, as IBM embarked upon transforming itself to a “cloud first” business, it conducted an industry study on what was being done regarding DevOps. Its DevOps journey entailed the following:

- 1) Elevating the status of operations engineer as a discipline in the DevOps world.

- 2) Characterizing DevOps as a practice and way of doing things; not an organizational construct, which perpetuates notion that someone else is responsible. DevOps encapsulates the entire organization in which everyone has a role.
- 3) Aligning development and operations under a central operations leadership team, with the entire development organization owning the operational discipline to ensure proper delivery.
 - a. DevOps was built to address the misalignment between development and operations. If the development director did not own the full life cycle of the deliverable, the organization would lose alignment.
 - b. Success hinged upon a proper business alignment throughout all disciplines
- 4) IBM is historically an enterprise software delivery company.
 - a. While it performed well at designing, building and testing code, deployment and monitoring fell to the customers, making it difficult for the engineering teams to accept responsibility.
- 5) People are the biggest technical problem: i.e., effecting cultural change that embraces and conducts DevOps correctly. Efforts included:
 - a. Buy-in: Ensure that all agree on DevOps and that the fundamental tenets of DevOps are worth pursuing
 - b. Realized value: Small successes realizes the possibility of doing things differently.
 - c. “Generative actor”: Identify and align leaders who will lead the organization and those who hinder progress. Consider the impact of negative culture on the DevOps journey.
- 6) Empowerment of Individual Developer: unlike a typical enterprise mode of software delivery, individual developers are empowered to make decisions about code going into production; this fundamentally changed what they did and how they acted.
- 7) Cloud-based solution: As Cloud starts small, must keep costs down to make a profit. Enterprise software delivery is “high touch” delivery: Cost is measured by the number of folks, to number of times they had to touch servers. Cloud is low to zero touch.
- 8) Operator-to-server ratio to measure cost. Can’t survive in the cloud business at a high ratio.
 - a. Major global cloud-based services operate at a high ratio
 - b. 1 to 20K: can’t operate at that scale
 - c. Infrastructure service providers (1 to 2k+): good ratio, but still high touch
 - d. Average Enterprise (1 to 100s): still high touch and can’t survive in cloud business.
 - e. Limits of taking Enterprise architecture to cloud forced bigger changes
- 9) Resiliency of system to failure is important; not just shooting for reliability. Ability to improve time to recovery; i.e., returning services to customers is more important.
- 10) Dark Launch policy: Deliver features and enable them separately from delivering code. Huge risk mitigator that improved the quality of the deliverable.

DevOps - Nitin Madhok

Definition of DevOps:

- Originated in 2008 during a talk on Agile infrastructure by Andrew Shafer & Patrick Debois.
- Breaking silos between teams to work more collaboratively using CI/CD automation tools.
- Methodology with certain methods, values, principles and practices and the tools to achieve them.

- Practice/methodology: DevOps aims to unify teams and break silos. Not simply combining Dev and Ops: shorter development cycles, increased deployment frequency, with the goal of more dependable release. Providing platform, tools, knowledge and resources for teams to work better together and bridge the gap between them.

5 Goals:

1. Continuously develop and test code.
2. Continuously integrate code from other developers into main code.
3. Continuously deploy using automation tools.
4. Continuously monitor code for failures (revert quickly).

DevOps is Not

- Not a Single term/process
- Not the Development side trying to eliminate operations; operations have not kept pace with development team. Will end up automating tasks if operators don't do it.
- Not only CI/CD tools. Knowing how to use tools is as helpful as the methodology itself.
- Not just a role/job title. Important to adopt principles and titles
- Not just "Dev" and "Ops" Security, management, network teams, DBA teams collaborating.

Why DevOps?

- Quick on-time delivery with no impact on quality; gives more scope of testing at end of each phase.
- Faster development cycles: Using CI/CD tools and by engaging developers earlier in the process, code is continuously tested as it is developed.
- Improved collaboration: since teams work together, more opportunities for exchanging ideas and teams can work together for a common end goal.
- Increased productivity and efficiency by automating most tasks
- Reduced cost: lower failure rate and increased productivity and efficiency due to automation, faster deployment.

5 DevOps Challenges

1. Culture: build collaborative culture with shared goals; hire, support DevOps champions
2. Automation: Important at all levels. Focus on test automation and consider security.
3. Legacy systems: include modeling legacy applications - difficult to install new hardware
4. Application complexity: need to be decided early in process. Currently, applications are designed to keep the flexibility of containers in mind.
5. Skillset and tools: teams need training on DevOps methodology; culture enterprises should avoid fragment skillset and options

Leveraging Containerization for DevOps with Sandia's HPC Workloads - Andrew J. Younge

A pragmatic viewpoint on our current view on DevOps and bridging MAGIC's conversation on containers.

Motivation

Containers have changed how DevOps is done in the industry/cloud; large shift in cloud and industry space. Specific Impact on integrated code teams' development for large scale HPC applications?

- Roadblock: Docker and Kubernetes are not a reality in these systems. Development is fundamentally different because building large applications with many interrelated dependencies which scale to thousands/tens of thousands of nodes, not micro-services.
- Performance and security remain a priority, particularly with some of the existing workloads

Can Containers help with applying DevOps to HPC integrated codes?

- Start on laptop and scale to production Supercomputer Facilities, while using other resources along the way? Will it provide software development mechanisms inherent in the model?

Container Features wanted in HPC

- Key concept: Ability to Bring Your Own Environment (BYOE)
- Composability and Portability: enable an exacting definition of build-specification for third party applications and libraries. And ability to move to another deployment or deployable system.
- Version Control integration: work within and leverage as best as you can.

Not wanted in HPC:

- Overhead: can't sacrifice performance at scale
- Micro-services. They are batch-based applications
- On-node partitioning: pack many containers on one node and consolidate resources in cloud for efficiency
- Root operation (nonstarter)
- Commodity networking: not working with these tools; not interested in TCP/IP.

Container DevOps Vision at Sandia

- Enable true software development on individual developers' laptops, provide consistent development environment with local and consistent code to speed up development turnaround
- Let developer specify application and environment to run on because developers know best. Consistency from day 1 and throughout is important.
- Can work with different containers and maintain branches where each HPC application can focus on core mechanisms
- Make a cohesive development environment and something that can move to a production supercomputing facility (often challenging) E.g. Cray Compiler/tool chain cannot run on laptop but it is a key functionality for utilizing and developing some production HPC codes. Would be nice if users can import container and run on a target platform.
- Reproducibility: May make some aspects more tangible.
- Pay attention to different architectures and different compilers because codes need to run on diverse supercomputing hardware.
- Let Developer specify exact run-time environment, and steps on how to build code. Then can hand off to new developer and potential delivery mechanisms to analysts who may be running these codes on production resources
- Can integrate with new CI tools and allow rebuilds to occur (consistent environment to run unit and integration tests on potentially different sites – still being worked out)
- Move from laptop to supercomputer.
- **Trilinos Muele Example** (simple container example): Shows steps for installing and building Trilinos.
- Large-Scale HPC application: Impractical to go to supercomputer to do basic function test. Did code fundamentally change how the application will run?

- Will sit for days in queues; problematic for enabling turnaround time. In past, bought smaller versions of supercomputer for specific development: just to have cohesive environment from your production resource and where doing development.
- Model that we are currently putting together alleviates some of these problems. Provides a container with your exact development environment (initial testing and builds on laptop) and moves to separate system; Leverage Gitlab container registry services (can sit on different networks and move to deployments). With HPC container efforts, can start to run on production resources (applies to singularity shifter, and Charlie cloud).

Discussion

- DevOps perspective for HPCs differs from perspective for Cloud resource because building large integrated codes; but containers can still play a role.
- DevOps model for custom HPC ecosystems (e.g., build with Docker on laptop (do unit tests, etc.), store container in registry, ultimately, ship to supercomputer)
- Performance can be near native if do it right. Within ECP container effort, working on best practices to figure out how to do this consistently and ensure that it won't lose performance
- Expanding this model for some integration teams; new arm-based supercomputer

Future Directions

- Security auditing processes remain an open question: how to get facilities to approve of an entire container instance moving towards more secure networks and facilities? How will the facilities ensure that these codes do not change over time. Opportunities to provide consistent model of delivery of software that can be audited and validated.
- In the past, there was a large divide between development and the system administrator for developing HPC applications and running them on various systems. The divide is shrinking and they are working together more; need to iron out processes of working together.
- More work to be done to support from systems software and architecture perspective
- How does this look when start to look at emerging software ecosystems (e.g., non-batch based streaming workloads)? Can start borrowing and incorporating models from the cloud).

Innovative DevOps: Tom Morton

Background:

- National Defense Strategy: DevOps and agile technologies (cloud) fit into the strategy of reforming the DoD for greater performance and affordability. DoD aims to improve its performance to develop and deliver capabilities and intelligently apply new ideas to DoD's primary missions. DoD is comprised of agencies working asynchronously across a range of DevOps; challenge is to pull together communities involved.
- More than "Dev" or "Ops": DevOps is multiorganizational with multi stakeholders. Collaboration is needed between many parties for continuous deployment. E.g., DevOps-like engineers who can run infrastructure/pipeline but must work in policy, governance, accreditation, network monitoring

DevOps Culture/Mindset:

- Continuously deliver performance with affordability and speed as we change departmental mindsets, cultures and management systems. DevOps is not just about tools or automation;

rather it is about cultural change that needs to be embraced by all teams (program management, cyber operations, etc.).

What is culture? And how do we change it?

“Innovation is the adoption of a new practice in a community.” *The Innovator’s Way, Essential Practices for Successful Innovation* by Peter J. Denning & Robert P. Dunham.

- Interpret in the context of bringing DevOps to an organization,
 - E.g., DoD. See also Apple Company: Steve Jobs was able to change the practice of a community by implementing a vision of changing the world, 1 person/1 computer at a time.
- First way to change practice of community is to successfully engage with others about an idea and grow it.
 - E.g., CTO: begin continuous deployment with 1 team; got commitment from agency director to try it, which was successful and received more commitment from boss; grew to 5 applications to achieve continuous deployment to continuous accreditation (means, delivery pipeline deploys to production environment).
- Build upon your success (adoption) – become more innovative in driving adoption of ideas you bring to your organization.

Discussion:

What science research use cases are driving the need for DevOps?

- Spectrum of users: sort out communities within infrastructure users and providers. Much of the user community doesn’t develop code that would be recognized as software development, but they develop tools, scripts and abilities which can create many problems. We still need to manage automation in the least sophisticated user community. When changing how an infrastructure is deployed, consider that the spectrum of users extends to the least sophisticated – creates dilemma when approaching DevOps (who are we discussing – developer of workflow tools, HPC?).
- If we believe in true distributed systems, we have no choice but to do DevOps because everything discussed today is inherent to a distributed system. Whatever we do will end up becoming DevOps; necessitates a different approach. Must be able to quickly deploy and undo a mistake.

Thoughts on updating and managing distributed system? Anything in research or from a new DevOps approach?

- Context: Using DevOps in an environment where have embedded systems or IoT (commercial case) and managing it in a homogeneous way despite possible heterogeneous components at the edge of infrastructure.
- Sandia: Diversity of Supercomputer perspective. Problems are potentially similar (e.g., need to enable integrated code team to run on an arm-based Supercomputer). Many active questions and research opportunities to best define how DevOps model will work. Some practices may apply to IoT; may get at performance portability questions. How do we deal with heterogeneity in this model?
- High performance computing is one instance. Interested in relevant work regarding the automated software lifecycle model in the distributed, embedded legacy area.

- E.g.: Open Science Grid (OSG) is an operating example as computers are scattered globally (some have latest GPUs, others do not, and it all works). Distributed computing has been overshadowed by grid, cloud, IoT, but at its core, heterogeneity is a distributed system concept.
- From cloud or embedded supercomputing perspective, many provisioning strategies have occurred. DevOps promises the ability to code once, distribute, and attain a collective result. Interest in looking at traditional methods of provisioning and figuring how to map it to the ability to code once and distribute everywhere.

Focus on ability to arbitrarily control the version currently using.

- This has not always been the case with deployed distributed infrastructures. Notion of DevOps that allows the user to be part of the continuous nature of the cycle. Distinguish among broad spectrum of developers and users: community is comprised of those who run OSG clusters or users of those facilities?

Security is Interlinked with development and operations:

- Development, operations and security are interlinked in the modern paradigm. “DevSecOps”: everyone thinks about security continuously throughout the chain. Trusted CI folks are thinking about it.
- Security is needed for successful deployment; define everyone’s role in each aspect of development, operations and security.
- Need to give users tools.

OSG “software assurance marketplace” DHS-funded project

- Based on our notion of continuous assurance: do it earlier, do it often. If there is interest, OSG can give a status of project – have OpenSource software designed to bring static software scanning capabilities into software development because in order to get high quality software out quickly, it is important to have a unified framework to continuously do software assurance.

MAGIC CY19 tasking

CY18: Evolved over winter Containerization. No need for containerization workshop. DevOps workshop open for discussion.

CY19: Need to be finalized by October meeting at the latest.

Format: Reached consensus on a mix and match format: 1-2 multi-session series and one-off topics. Will modify approach as community sees fit. Include speakers from outside of the community.

Potential topics:

Cross-agency Activities. Identify what can be done across agencies and connect silos. Can we do better across agencies? Seek out successful projects (e.g. NSF’s Office of Advanced Computing workshop: one panel about international and interagency opportunities in research computing infrastructure). Potential areas of interest:

- Software development techniques (e.g., OSG’s DHS continuous assurance project);
- Networking level
- Distributed grid

Action item: think about how to implement the idea

Data Gathering. Coming from the network world, when starting to debug network problem, the infrastructure needed to gather data from belongs to someone else. So, it is difficult to obtain data. These same problems exist as we move into more distributed computing environments with many workflows; there are no good tools for locating the problem and reporting it to support staff who will act upon it. Perhaps we could shed some light on this scenario and find ongoing work in this area.

Discussion:

Needs to be edited.

Contact RDA and possibly digital library community.

Explore interrelation between software and data. See Dan Katz's work on software citation.

Provenance issue and verification. As we are using different machines, is there something more foundational, beyond the current, ad hoc solutions, for tracking data?

Discussion:

Motivation: Managing complex infrastructure. Creating without ability to own either single piece or in its entirety.

Quality assurance and provenance related work is a big deal in data archive; relates to MAGIC group.

Edge Services. In order to make entire spectrum of infrastructure useable with different flavors of grid and cloud, etc, we are spending much time in developing solutions or finding ways to interoperate heterogeneous set of resources is with "edge services" (sits between interface provided by resource and the application using for scientific expertise), formerly middleware. What are these edge services and what do we need to make the entire infrastructure useable for the scientific community? We are having to develop real services to make resources useable. Joyce Lee will follow up with Kaushik De.

Academic community

At last month's meeting, we discussed having more involvement from the academic community. Curricula changes tied into workforce development was noted. It was suggested that the academic community can provide updates at monthly meetings. Joyce Lee will follow up with Don Riley.

Upcoming Events and Activities

- Coalition of Advanced Scientific Computing Report
Gathering material on ROI and the academic use of cloud computing. Return of financial grants and academic returns on how to deploy cluster and cloud computing). Topic of distributed computing collaborations will likely be raised. Alan will solicit input for topics to flag in the report: How to call out efficiency of distributed computing collaborations such as OSG and technological developments that can minimize the friction or difficulty for people moving from university-based clusters to national scale resources. These impact ROI when extend to return on time and effort on behalf of researchers. Discussion will be held at PEARC 2018 led by Alan Chalker in July 2018.
- SC18: MAGIC will hold its meeting on Wednesday, Nov 14 from 1:30-3:30pm (Room D175).
- DevOps Speaker series:
Need names and speakers from Operations, Research, Scientific user community. What else we can do in this area?

Next meeting: August 1 (12 noon EDT), National Coordination Office.