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

Middleware and Grid Interagency Coordination (MAGIC) Meeting Minutes
September 5, 2018, 12-2 pm
NCO, 490 L'Enfant Plaza, Ste. 8001
Washington, D.C. 20024

Participants (*In-Person Participants)

Richard Carlson	DOE/SC
Kaushik De	UTA
Leo Garciga	DoD/OCI
Venkat Kodumudi	CGI
Padma Krishnaswamy	FCC
Marshall Lamb	IBM
Joyce Lee*	NCO
Brian Lin	UW-Madison
Thomas Morton	DoD/OSD
Sean Peisert	LBNL
Matyas Selmeci	UW-Madison
Alan Sill	TTU

Proceedings
This meeting was chaired by Richard Carlson (DOE/SC) and Rajiv Ramnath (NSF), who will be rotating out of NSF at the end of September 2018.

Speaker Series: Operational Challenges in DevOps

- *Operational Challenges involved in implementing DevOps*, Leonel Garciga, Chief Technology Officer, DoD Defense Threat Reduction Agency, Joint Improvised-Threat Defeat Organization
- *Perspectives on DevOps at CGI*, Venkat Kodumudi, Director, Innovation & Outreach, Emerging Technologies Practice, CGI Federal
- *Enterprise to Cloud: Operational Challenges*, Marshall Lamb, Distinguished Engineer and CTO, Watson Supply Chain Watson Customer Engagement Division, IBM Corporation

Speaker Presentations
Operational Challenges involved in implementing DevOps - **Leo Garciga**
Focus: Operational capability (multi-year effort)

1) Layering and building out ecosystem: Look at entirety of ecosystem (what it does), not just DevOps, but the bigger piece that it needs to support
2) Build for Security
3) Build for Speed and Quality: build software both quickly and securely onto a production network

Eco-system piece is important

- Usually 18-24 months to transition to DevOps; shorter timeframe if already using Agile to deliver software.

- Operational
    - DevOps pipelines for pushing out capabilities are not all created equal. There are many common tools that folks can use (source code repository, security tools). Ensure that you have the right capability from the start when looking at the actual software being developed and what it's being deployed to.

Building for Security piece (cultural piece). Typically, it is an organizational challenge to shift mindsets and ease into this mode of operation.

- Trying to leverage automation and testing within their pipeline and adding cybersecurity at the end of a delivery cycle is a big cultural piece to break.
- All tools, security checks, and automation are based on building out specific thresholds and checks for software to be deployed against the network.
    - Should have a good idea of your eco-system security posture, capabilities security posture and risks which they are willing to take.
    - Federal agencies: integrate authorizing official into process who has the skillset and thorough understanding of setting thresholds.

Building for speed and quality

- Focused on cultural piece to get there.
- Key issue: Quality
    - We increased quality by moving fast and having the ability to pull capabilities back quickly. This drove how we did business from beginning to end. Transitioned to Agile quickly (redid policy, process and retooled workforce).
    - Went through transformation again in the move to DevOps. In the process, we built a more transparent approach to governance, most of which was automated.
- Slide depicts approach towards new technologies and ways of pushing out capability in a governed approach. Setting thresholds is still challenging.
- Note the networking side of problem: Incorporate sight reliability engineering piece into the pipeline from a capabilities perspective. Ensure that the network is supporting it.

***Perspectives on DevOps at CGI* - Venkat Kodumudi**

<u>What DevOps Means to CGI?</u> (Slide 2)
Need to look at DevOps in conjunction with Agile.
DevOps touches upon all aspects of software development life cycle. (Slide 2)

<u>DevOps reference model</u> (Slide 3)
We built a model as a starting point (Slide 3). View it as 4 pillars and need culture to permeate in every process to change the DevOps practice.

- Governance and Organization (life cycle needs to work together to ensure success of DevOps)
- People (need consistent and common collaboration across teams)
- Process (needs to change)
- Technology (where try to automate or choosing right tool stack)

<u>5 pillars of DevOps Principles</u>: Drive DevOps and ensures its success (Slide 4)

- Software Dev standards- manage and ensure standards are maintained
- Quality Management and Automation – where automate different aspects of development
- Business and IT Alignment - continuously test or introduce model concept of acceptance (Acceptance Test Driven Development)

- Full Stack Teams- Agile teams that can do everything (e.g., developing story, or Jenkins cookbook, or Chef)
- End User Engagement

DevOps Best Practices (Slide 5)
Culture – need to enable cultural change
- Disruptive – because touches every aspect of DevOps area, from people to technology
- Governance- Tools that assist in automating the governance are the most important. Can focus on the mapping between Agile and DevOps; then able to see holes.

People - Need different mindset
- Collaboration needed
- Soft skills Development

Changes in Integrated Processes
- Need to change. In Federal government, agencies fear losing control once it is automated.

DevOps Tool Set
- Most existing tool sets are open source. Need to decide upon tool sets ahead of time, depending upon requirements.

Identify Trusted Team with Mature Services

Use Cases
Center for Medicare and Medicaid Services
- Introduction of DevOps and Agile resulted in about 20% reduction in defects. 50% reduction of life cycle time required for changes to get into production.
- Reduced deployments to 2-3 deployments/week; Used to be a 6-week cycle between deployments. Remaining issue: manual pushing of button. Can deploy with little or no down time.

Another customer
- Reduced testing cycle from 4-6 cycles to 15-30 minute window. Could reduce and control cost of the production environment in the cloud by using automation and DevOps principles to turn off production environments overnight.
- Could automate and scale up environment to increase volume without any down time or customer awareness that we were doing it automatically.
- Used fail-safe methodology to detect and fix issues early in the cycle and reduce costs.

Enterprise to Cloud: Operational Challenges **- Marshall Lamb**
IBM Connections Cloud: 4 challenges faced when transitioning from enterprise team to delivering a SAAS product. Deep in enterprise type customers.

Challenges facing companies with heavy enterprise-centric customer background (Slide 2)
- Separation of Duties
- Monitoring and KPIs
- Issues with long-lived systems
- Enterprise architectures- why unfit for cloud

Separation of Duties: Unsustainable for DevOps (Slide 3)
- Traditional model to operate a cloud solution: developers couldn't directly change production and operations couldn't change the code. Difficult to operate the solution by looking over operators' shoulders

- Modern interpretation
    - Proper Access control: to protect integrity of system and data. Allowed developers read-only access. No access to customer data or storage devices- but to log data and spec configuration. Drastically reduced incidence response time.
    - "Operations as Code": In DevOps, much of what we are trying to accomplish from operations perspective is writing automation code. Operators are changing code (automation and delivery pipeline code) in ways that directly impact production.
    - Change operations practice to follow DevOps practice – Separation of duties pertains more to the strength of the delivery pipeline (talks about type of testing, nature of testing and extent of testing performed in the pipeline and who has the ability to control it). Test teams responsible for building test harnesses, generate test cases and right checks and balances in pipeline.

Monitoring and KPI (Slide 5)
Due to separation of duties, only operators had direct access to data, but other stakeholders (operating managers, developers, etc.) did not have firsthand access to data needed to make informed data-driven decisions; no visibility to that.

- Game changer: Removed from data center and made data available to all stakeholders. Then started to identify and quickly address problems that were previously unknown. Then we were freed up to address other problems related to KPI metrics (data science, anomaly detection, and end user instrumentation)
    - Challenges: operations feared allowing developers' access the system would lead to questions regarding operations' practices. Framed it as partnering with operations to improve the system.

Systems as Pets: Long-lived systems (Slide 7-9)
History of treating systems as pets, instead of rebooting and plagued with related issues (configuration drift, memory leaks, runaway processes)

- Regular rebuild methodology (in place of fundamentally changing how took care of systems)
    - 17 days of memory leak. Traditionally trained to quash these leaks. In modern cloud environment, no server should live more than 17 days, now in container-based environments, often servers don't live for more than a few hours. Once get into habit of constantly restarting servers and rebuilding, address 60% of customer impacting issues
    - Focus on continually re-installing software. Install and test software and swing traffic to it (A/B flip approach for side A and B of data center).

Costly Architectures (Slide 10)

- Spent years improving our system and trying to drive down costs of running our system. Measuring cost in cloud: How many human beings touch how many servers how many times. Human factors are highest cost in maintaining a system. To improve cost structure, automating highly manual tasks plays a big role.

Operator to server ratios (Slide 11)

- Major global cloud-based services
- Cloud platform and service providers
- Average Enterprise: poor efficiency and very costly; IBM was here. What operational challenges did we have to fix?

Painful Realization (Slide 12)

- No ROI if invest in automation because software was high touch (requiring manual changes). So put some resources on higher value work and also started over in some cases (used cloud corporate technologies from beginning, instilled right DevOps culture and got it right).

Discussion

KPI monitoring

- Need transparency as you build the software (from requirements owner to entire staff).
- To reach the point of Operational secure DevOps pipeline and process, need to make serious modernization decisions up front.

Greenfield development and designing it to be automated.

- Designing apps for security in mind does exist with or without DevOps as part of designing standards. On flip side, use sec and app testing tools to ensure that these standards are met.

DevOps implementation today

- Focused on baking in security piece than anything else. Different from build-code for security, rather building within your framework:
- There are upfront decision points by ultimate decisionmaker where mitigate before you write the code. Not fully automated due to technology gaps, but close.

Theme of group

- Common infrastructure among large-scale labs, universities, middleware projects. Operate large-scale infrastructure for science and other mission uses throughout federal and research enterprise.
- Wish to reach out to folks who focus on this area while addressing security (e.g., NSF Center for Trusted Cyberinfrastructure).

Updating software

In running a typical cluster, deployed infrastructure for stability reasons and software reproducibility; both users and administrators tend not to update their software. Continuous integration and deployment etc., is antithetical to how many view infrastructure software where stability and software reproducibility is paramount. With these concerns, how do we introduce such ideas into large-scale infrastructure?

- Example: applied forcing function that instilled proper behavior:
    - Containerization strategy
        - Container prevents you from doing anything productive inside the running operating system inside container.
        - Forces the building of more rigorous automation system around it to get at information to do log collection and correlation, monitoring on the outside and no notion of software updates (can't push into a running container)
        - Only way to get update is to deploy a new container. Focus on lifecycle of software as a function of continuous installation with no updates.
    - Rule: No human is involved in software deployment. Using containers, no one is running an install command. So forcing developer to ensure first and only time that software is installed in a container is right after the build and container, not software install, is propagated. Operators and testers are forced out of the pipeline; thus, forced a higher degree of automation earlier in the pipeline.

    - Transparency: Historically, product owners take a long time to discover and fix a problem. Shortened time it takes for software to get pulled back and re-deployed. But level of transparency provided is key. Once they had real time visibility, easier to get product owners on board because knew they would have to take some programmatic action.
    - Containerization is one way. There is no full answer yet and it is a growing concern. Sometimes, automating governance will help in catching problems ahead of time (security, gaps).
- Is there a maturity model, with associated metrics, that organizations seeking to move to DevOps can measure their progress along the way?
    - Group closely associated with IBM team that has done something like that. Rajiv Ramnath will reach out to him. Scientific research organizations should strive towards some level of DevOps.

CY19 Tasking Discussion

CY18 Tasking
Containerization and DevOps series, possibly workshop.

CY19 Tasking
- Data life cycle series (4-5 months); includes provenance
- Single topic issues (identity management, edge services, academic community)

Compare data life cycle to software life cycle
- As we look at cyberinfrastructure program and move it towards more integration, how is the data life cycle different from the software life cycle.
- What should be proposed regarding life cycle approaches to data vs. life cycle approaches to software?

Data life cycle moving to future
- Need to add distributed nature of future data. Ensure data isn't viewed just as individual silos. Data life cycle is made more complicated by the amount of data and its distributed nature.

Streaming data
- Variety of long standing tools for data storage and transfer (e.g., dCache, Globus, Research Data Alliance, data transfer project). Morris Riedel (European projects)
- Also, reach out to new projects data transfer project (collaboration of major cloud computing players focusing on building a common framework with open source code) –

Annual planning meeting
- Agreement: multi-month series on data life cycle as it moves from local storage devices to distributed environments with instrument data, archival data, re-using data, knowledge bases and other new things that will be required to analyze data (analysis tools, ability to move, find and use that data; Controlled Unclassified user information).

Workshop
- Determine later as it will be influenced by the sessions.
- Planning takes about a year. If we do it now, probably September/October.

Academic Roundtable

- Will be added for awareness. Rich received an email invitation from Coalition for Academic Scientific Compution (CASC); can speak about MAGIC. Focusing on data will be timely as finishing work on ROI. Turning to community collaboration.

Roundtable

October 3-5, CASC meeting. Rich Carlson has been invited to speak.
November 11 – 16, Supercomputing18, Dallas, TX
November 12, SC18 Data center automation workshop will cover data center automation technologies, from server control to provisioning systems. Discussion will address what is automation that can help with security. Recruiting topics and speakers by September 17.
November 14, MAGIC meeting 1:30-3:30pm CT, Room D175. Will be identity management from a global perspective.

Next meeting:

October 3 (12 noon EDT), National Coordination Office.