DEVOPS

Transforming Software Development

Jimmy Pham / Haluk Saker
Booz Allen Strategic Innovation Group (SIG)

March 2016
Agenda

Our DevOps Point of View

Our Experience

Booz Allen DevOps Service & Capability Offerings

The DevOps Practices

Delivery Pipeline
DevOps is part of our Modern Software Development practice

Our DevOps Is The Culture, Practice, and Automation of Continuous Deployment Unifying
Agile Development, User Experience, and Security

- Team Owns All Aspects from Dev to Ops
- Treat Operations as First Class Citizens
- Development Acts as First Responders
- Shorten the time for Resolution of Issues
- Shorten the time for Delivery of Features
- Guarantee Stable and Repeatable Operating Environments
- Minimize / Automate Coordination to Deploy & Deliver
- Bake Security in from Dev To Ops
- Proactively Stop and Fix Potential Defects
- Become a learning organization with full transparency & continuous feedback
DevOps is as much about the people and culture as it is about the technology and processes

**People:** build teams that embrace the principals and culture of DevOps

**Process:** define/redefine processes to reduce steps in delivery and increase speed by automation

**Culture:** change the behavior and mindset to establish a culture of transparency, joint ownership, and continual improvements

**Technology:** choose technology that enables automation, visibility, traceability, and integration

Source: Gartner (2015)
Agenda

Our DevOps Point of View

Our Experience

Booz Allen DevOps Service & Capability Offerings

The DevOps Practices

Delivery Pipeline
We pride ourselves in staying ahead of the technology trends and practices and being able to help customers realize their development and delivery goals.

A subset of our partnerships, technology expertise, and customers:

<table>
<thead>
<tr>
<th>Partners &amp; Technologies</th>
<th>Industry Presence</th>
<th>Federal Customers (DevOps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>amazon web services™</td>
<td>docker</td>
<td>U.S. Citizenship and Immigration Services</td>
</tr>
<tr>
<td>docker</td>
<td>SxSW</td>
<td>GSA</td>
</tr>
<tr>
<td>red hat</td>
<td>Chef</td>
<td>CMS</td>
</tr>
<tr>
<td>Chef</td>
<td>GitHub</td>
<td>HealthCare.gov</td>
</tr>
<tr>
<td>Google</td>
<td>Kubernetes</td>
<td></td>
</tr>
</tbody>
</table>

Spend less time building code, software stacks and environments, and more time building **mission/business value**.
Agenda

Our DevOps Point of View

Our Experience

Booz Allen DevOps Service & Capability Offerings

The DevOps Practices

Delivery Pipeline
As part of our Modern Software Development offerings we have several services around DevOps specifically

We can help you

+ Establish What DevOps Mean to You
  • We define 7 core DevOps practices and work with you to understand what they are and more importantly how they fit in to your current environment and how they need to align with your DevOps goals

+ Perform a Quick Look Assessment to Assess Your Current DevOps Maturity & Define Actionable Plan
  • We work closely with you your team to assessing your practices using our DevOps maturity assessment framework to help you determine where you need to be with each practice. Not every team would need the “highest” level of maturity. An actionable plan will come out of this assessment.

+ Define DevOps Adoption Strategy
  • As part of the plan an adoption strategy will be developed. Each organization has different barriers they need to overcome, it’s important to identify those success factors upfront and address

+ Design & Implement DevOps Delivery Pipeline
  • We will transform your software delivery technology and processes that reflect your DevOps philosophy, release strategy, and automation of your end to end development to delivery process
Agenda

Our DevOps Point of View

Our Experience

Booz Allen DevOps Service & Capability Offerings

The DevOps Practices

Delivery Pipeline
Continuous Integration: Code is regularly delivered to a code repository and builds and tests are automatically performed to find any issues. Alerts are sent if issues arise during the automated build process.

Configuration Management: The tracking and controlling of changes to the software code base and archiving of all file version into a central configuration management database. Provides organization for the software build process by accounting for multiple environments.

Automated Testing: Automated tests for unit, functional, security and performance. Testing occurs continuously throughout the development lifecycle, and provides insight into the current health of the software.

Continuous Delivery: Code continuously delivered to test environments and every software change is potentially deployable to production. Automated testing can drive the acceptance of software.

Continuous Deployment: All software changes that are accepted through automated testing are automatically deployed to production. These constant deployments do not impact end users.

Infrastructure as Code: Using software to control the build, configuration and deployment of the application. Additionally, software is used to control and provision the infrastructure of the system.

Continuous Monitoring: Ability to continuously monitor the system and applications for issues and then notify responsible parties in case of anomalies. Provide analytics gathering capabilities to ensure application is performing at optimal levels and the system is secure.
Continuous Integration

Continuous Integration (CI) is software development practice that requires developers to integrate their code into a centralized repository as they complete their code and unit testing, several times a day. The objective is to write the code in small chunks, make it work, and integrate it into the code repository as frequent as possible.

CI is the key agile practice that connects agile development into DevOps. It is a prerequisite for DevOps. Without well established CI practice, a development team cannot move far in the DevOps world.

Two main objectives of CI

1. Minimize Effort for Integration

2. Be able to delivery at any point
Often times we hear developers say “It works in my environment. I cannot reproduce the problem.”. This is mainly because the configuration management of the environments and source are controlled by manual processes and strict workflows are not necessarily enforced.

There are various complexities of source control workflow strategies:

- **Centralized Workflow** - single point of entry for all changes to a project

- **Feature Branch Workflow** - Isolated branches are added for each new feature work stream and this enables independent and loosely coupled designed principles. Feature flow also provides a foundation for Continuous Delivery and Continuous Deployment

- **Gitflow** – A more complex strategy that builds on top of feature branch but with specific designations branches for master, hotfix (patch), release, develop, and feature.
Automated Testing

For DevOps it’s all about automating as many test as practically possible. The results of these tests are the input to the team to take the next action.

Identifying the defects early and during the continuous delivery pipeline increases the confidence level to deploy the new capability to production.

The key is having full confidence about the automation steps.

Here are the list of recommended automated tests:

+ **Dev Env**
  - Unit Test, API Test, Build Script Test, Config Param Test, Infrastructure as Code Tests, Vulnerability Tests

+ **Test Env**
  - Function Test, Static Code Analysis, Performance Test, Load Test, Soak Test

+ **QA Env**
  - Smoke Test, Stress Test, Security Scans, Penetration Testing
Continuous Delivery and Continuous Deployment are two important practices that most times confuse teams. Continuous Delivery means that every change can be deployed to production once it is validated by automated testing but may choose not to do it due to business reasons. Continuous Delivery is the prerequisite of Continuous Deployment.

Continuous Delivery requires a cultural change. Team’s and customer’s expectation from the completed software needs to change.

Two golden rules of Continuous Delivery:

+ **Software (trunk) should always be deployable**: Trunk should be healthy at all times and ready to be deployed to production. It’s up to the gate keeper to deploy it once a day or multiple times a day. There should not be any technical difference. Timing and quantity of the deployments should be a business decision.

+ **Everyone should check-in to trunk from feature branches at least everyday**: This forces developers to divide user stories and features into meaningful smaller pieces. Every check-in is performed for the functionality achieved by team’s “definition of done”
Continuous Deployment

Continuous Delivery does not mean every change goes to production asap. It means every change is deployable and requires a business decision to move forward. Continuous Deployment on the other hand means the confidence level to the results of the automated tests are at the highest level and the change can be deployed to production if all of the tests pass.

Whether Continuous Deployment is right for the project or not is a business decision. Continuous Deployment is the ultimate goal for modern organizations. This does not mean that all of the capabilities would be deployed without any manual interaction. The key to automated production deployment is the confidence level for the quality of the software.

The factors that impact confidence level for going to Continuous Deployments include:

- Automated test coverage
- Automated test type (unit, functional, security, compliance, etc.)
- Multi-layer Code Reviews
- Product Owner Validations
- Advanced Deployment Strategies (Canary Release/Testing, Blue Green Deployments)
Infrastructure As Code (IaC)

IaC is another key practice of DevOps, to the extent that one can argue the recent maturity and adoption of tools and technology enabling infrastructure as code is one of the primary reasons DevOps has recently exploded into the scene with true end-to-end automation.

IaC goes beyond scripting for automation, your infrastructure configurations are treated just like application code going through all the best software practices of version control, unit tests, code, review, gating, etc. you can reconstruct your business as long as you have access to your version control system, data backup, and compute resources.

The goals of IaC should include the following:

+ Test Infrastructure in the same manner we test application code
+ Version control the infrastructure
+ Repeatability for tearing down and bringing up target environments in the same expected state
+ Self Documenting Infrastructure
+ Complete Audit Trail
Continuous Monitoring

Continuous monitoring is a critical practice that connects operations back to development. In the traditional IT operations, operations are performed by an operations team composed of system administrators, various tiers help desk, and analysts.

Continuous monitoring is the enabler of identifying the production problem and putting the problem into the hands of the developer to fix it quickly and reduce the time between identification of the problem and deployment of the fix.

Continuous Monitoring needs to include the following:

+ **Infrastructure monitoring** – visualize the infrastructure events coming from all compute resources, storage and network, and measure the use and check health of infrastructure resources.

+ **Application performance monitoring** – ability to target bottlenecks with the application’s framework. Quickly trace issues.

+ **Log management** – allows development and operations teams to collect logs in a standardized way and perform analytics to identify application and system problems.

+ **Security monitoring** – allows designing the system for security, achieve and keep compliance, to reduce security risk through automation. Security configuration management, vulnerability management and log intelligence to detect attacks and breaches early are achieved by continuous security monitoring.

+ **Alerting & Resolution** – useful alerting capability and defined strategy for resolution makes CM actionable
Agenda

Our DevOps Point of View

Our Experience

Booz Allen DevOps Service & Capability Offerings

The DevOps Practices

Delivery Pipeline
How do you automate and control a typical development environment that has multiple code branches...

<table>
<thead>
<tr>
<th>Environment</th>
<th>Configuration Management</th>
<th>Git Flow</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRODUCTION</td>
<td>MASTER BRANCH</td>
<td></td>
</tr>
<tr>
<td>RELEASE</td>
<td>RELEASE BRANCH</td>
<td></td>
</tr>
<tr>
<td>DEVELOPER (Integration)</td>
<td>DEVELOPER BRANCH</td>
<td></td>
</tr>
<tr>
<td>DEVELOPER (Local)</td>
<td>FEATURE BRANCH</td>
<td></td>
</tr>
</tbody>
</table>

1. Code Locally with Developer Branch
2. Continuous Integration of Developer Branch
3. Continuous Integration of Release Branch
4. Validate / QA
5. Continuous Deployment
...By Implementing a Delivery Pipeline

**What is a Delivery Pipeline:**
The set of tools and processes working together to provide the workflow automation reflective of your DevOps practice taking code changes all the way through moving it into production.

- Automation of building, testing and deploying
- Automation of Infrastructure which can be created and destroyed without impacting the health of the software
- Reflective of your release philosophy and strategy
- Repeatable with no surprises
- Visibility into the entire pipeline workflow
If tests are successful, push to your Docker Repo. Deployment of the updated Docker container is pushed out AWS ECS Task Definitions and any IaC was accomplished using CloudFormation scripts. Chef could also be used here to manage the container deployment and IaC.
CONTACT

Haluk Saker
Principal – DevOps Capability Lead
saker_haluk@bah.com
o: 703.984.0670

Jimmy Pham
Chief Technologist – DevOps Capability Lead
pham_jimmy@bah.com
o: 703.984.0646
m: 267.738.3805

Nirmal Mehta
Chief Technologist – Cloud Architect
mehta_nirmal@bah.com

Shums Hoda
Sr. Lead Technologist – DevOps & App Architect
hoda_shums@bah.com
703.984.1023