splunk> .conf2017

Dev/Sec/Ops

Building the Pipeline of Security driven process with Agile Methodology

Domnick Eger | Global DevOps Practitioner

08/25/17 | Washington, DC

Forward-Looking Statements

During the course of this presentation, we may make forward-looking statements regarding future events or the expected performance of the company. We caution you that such statements reflect our current expectations and estimates based on factors currently known to us and that actual events or results could differ materially. For important factors that may cause actual results to differ from those contained in our forward-looking statements, please review our filings with the SEC.

The forward-looking statements made in this presentation are being made as of the time and date of its live presentation. If reviewed after its live presentation, this presentation may not contain current or accurate information. We do not assume any obligation to update any forward looking statements we may make. In addition, any information about our roadmap outlines our general product direction and is subject to change at any time without notice. It is for informational purposes only and shall not be incorporated into any contract or other commitment. Splunk undertakes no obligation either to develop the features or functionality described or to include any such feature or functionality in a future release.

splunk> .conf2017

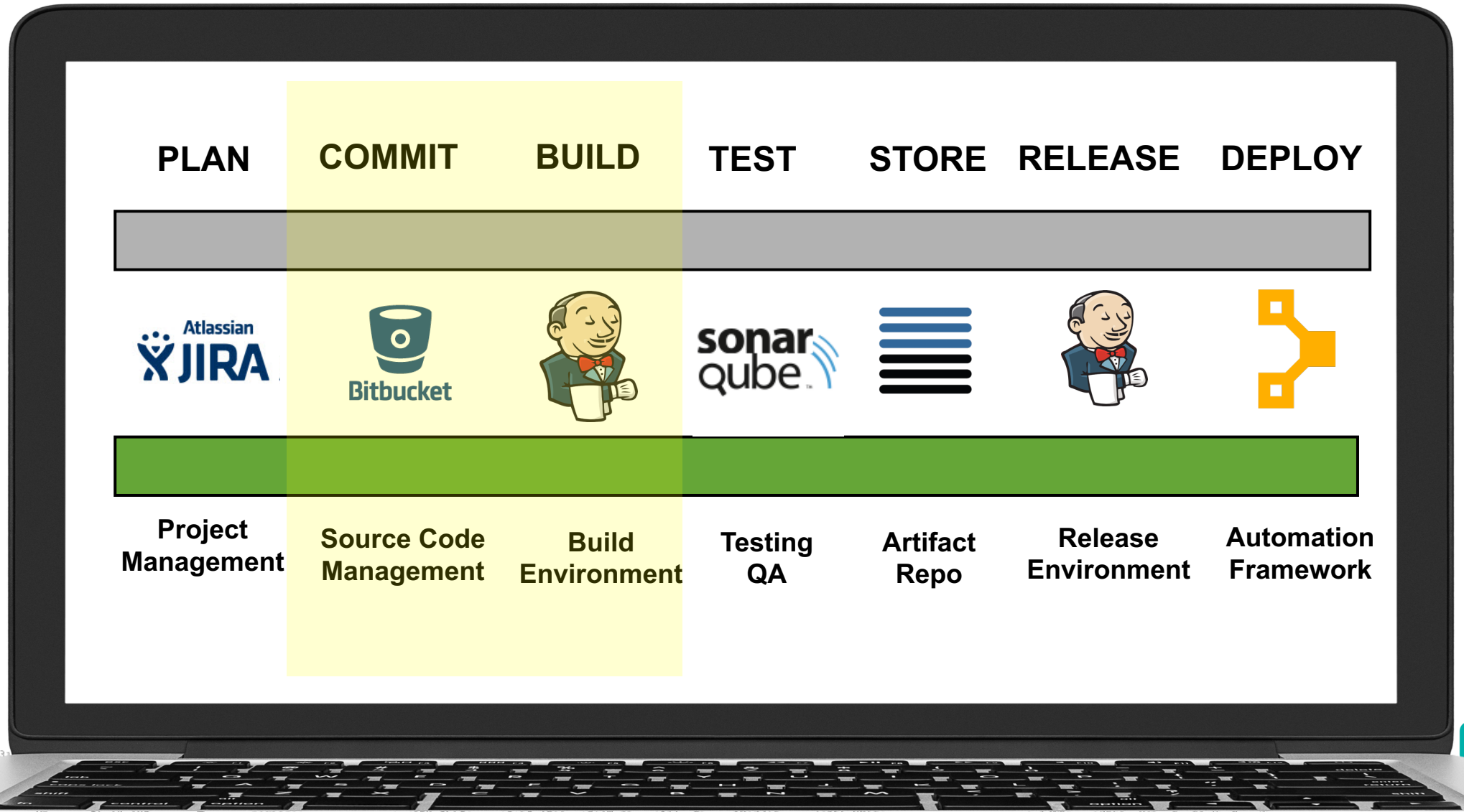
DevOps Pipeline

Gaining Visibility into the DevOps Tool Stack

splunk> .conf2017

DevOps Pipeline with a Security Focus

Plan, Code, Test, Store and Release



Code, Commit, Build, Exploit

Internal Hacker within the Ranks of the Development Team.

1. Actor Bypasses Code Control by Accessing Repo published accidentally as a Public Repo.
2. Actor Forces a Rebase and Pushes Exploit into code base. Audit trail is logged but no one notices.
3. Actor then pushes the code through the build environment and make the code available as latest artifact.
4. Code is deployed via Automated Webhooks and Is pushed to the primary repo for release automation team to push into production.
5. Actor has a backdoor and waits for the code to be deployed into the wild.

Act II

The Actor waits for the right time to strike.

Code, Alert, Secure, Respond

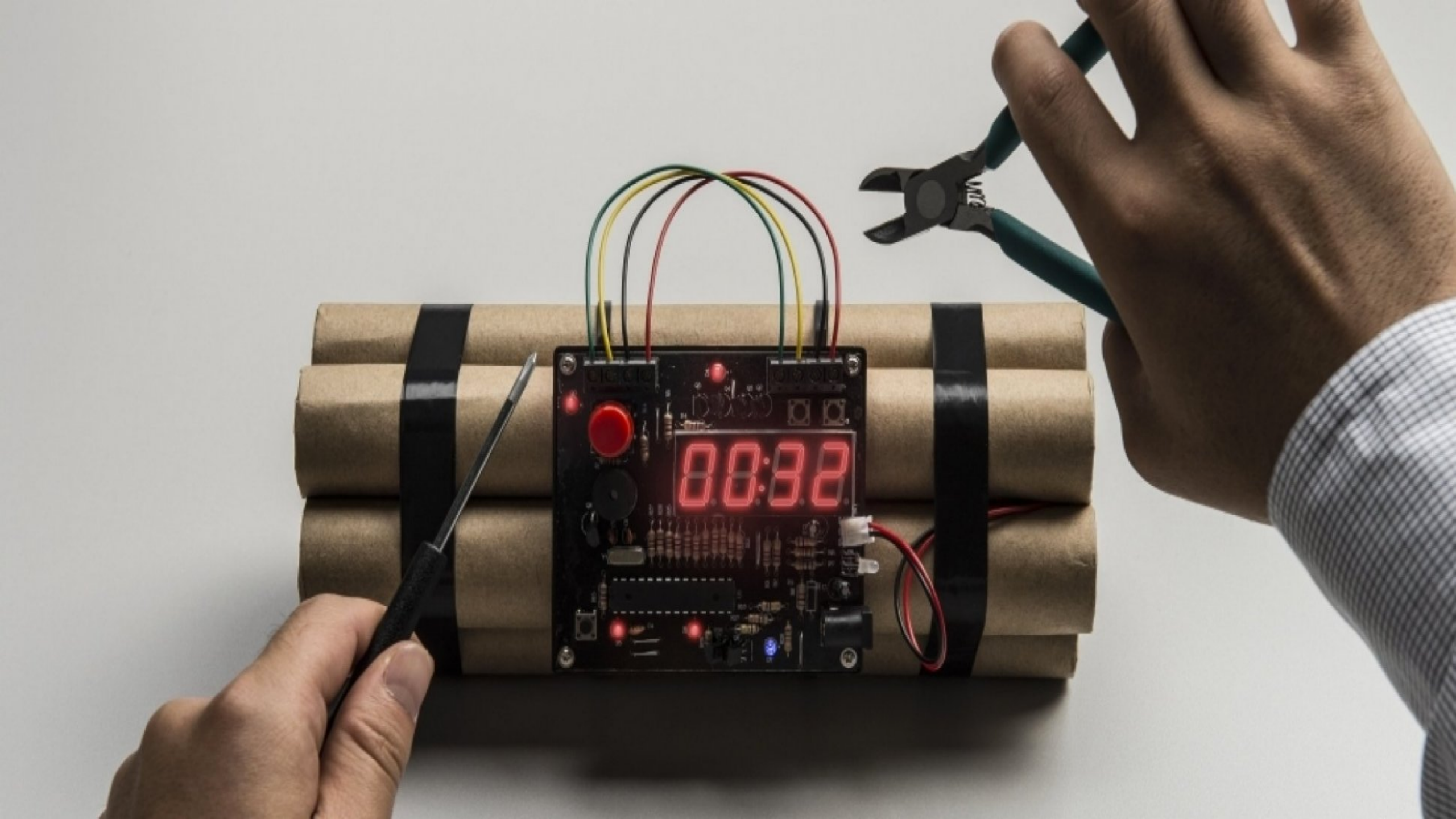
Taking the proper approach to auditing and alerting can help you save you from massive headaches.

1. Splunk triggers an alert to notify team that a repo is a public and not set to private. Splunk App for Bitbucket Notifies the Development and Security Team before Actor find the mistake.

2. Actor tries to bypass checks and get into Repo but can not rebase repo, tries to Create a branch build and waits for a pull request to be Approved. Developers have to go through two reviews before committing to master.

3. Actor tries to bypass code review method and clones the repo and changes the build plan which then trigger an alerts in Splunk and notify the team. Splunk locks out the account.

4. Information is logged, Actor is caught and everything in the environment is checked to validate code compliance.


00:32

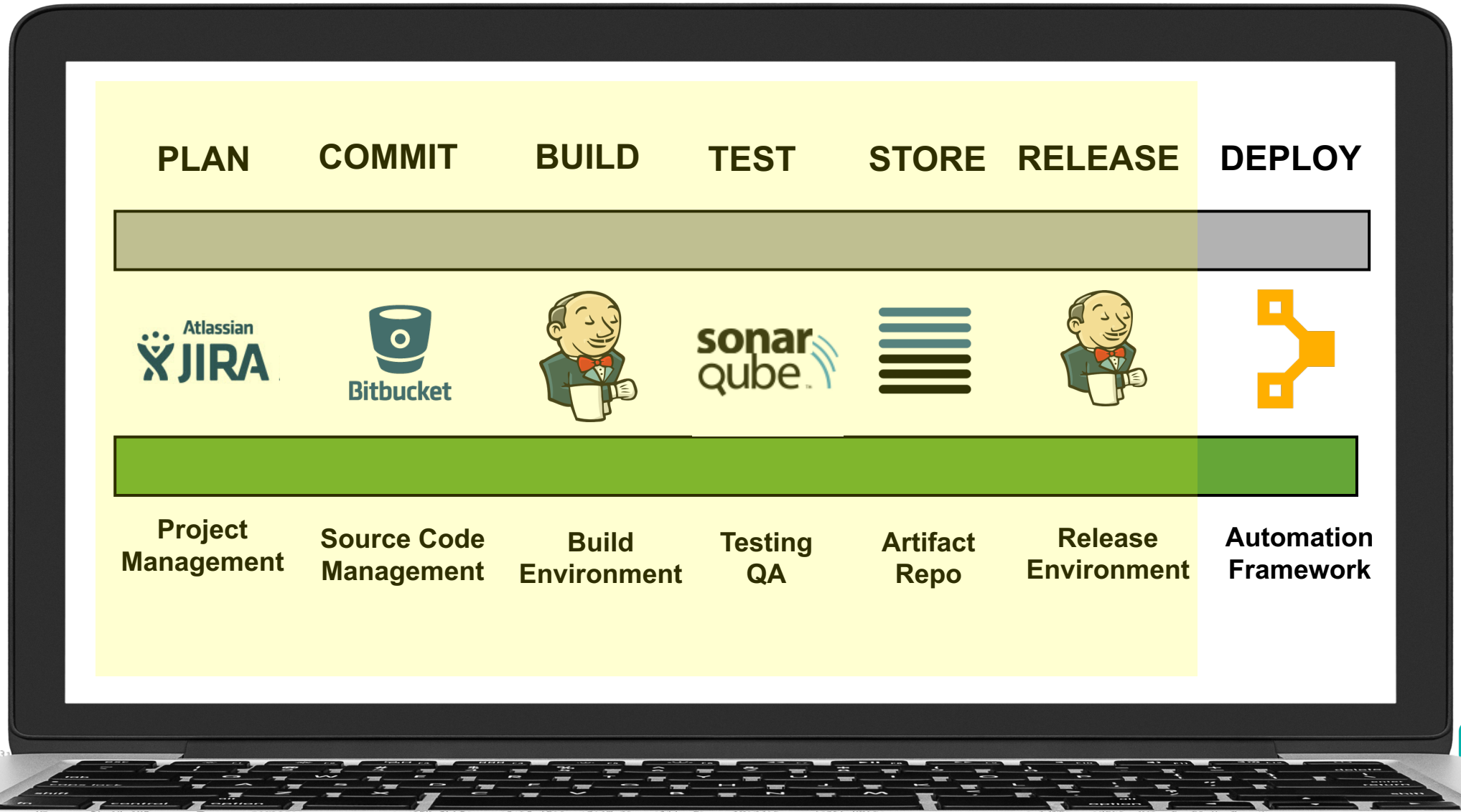
Planning and Executing

Following the Plan and Executing and Document the Process.

splunk> .conf2017

Planning and Executing

Plan, Code, Audit and Respond



Plan, Code, Test, Release, Backdoor

Hacking when no one is watching and building into the environment.

1. Actor attempts to bypass system checks and access backend MySQL Database to cover tracks.

2. Actor makes changes to the MySQL database, turns all Repos into Public and attempt a code exploit.

3. Actor attempts to bypass system by bypassing HIDS and trigger alert. But no one is monitoring the build environment that actively.

4. Actor pushes code, rebases the Repo, pushes past the code quality testing, forced artifacts to be created and pushes the final artifacts to be deployed.

5. Release engineering team blindly pushes into the environment and actor waits for the right opportunity to exploit code.

Code, Alert, Secure, Respond

Another attempt foiled by the logging all events happening in the environment and alerting the changes.

1. Actor attempts to bypass system checks and access backend MySQL Database to cover tracks. Splunk Stream detects a MySQL statement and logs the change.

2. Actor makes changes to the MySQL database, turns all Repos into Public and attempt a code exploit. Splunk get notified on the clone and push and keep it own record of the change.

3. Actor attempts to bypass system by bypassing HIDS and trigger login alert. Splunk notify the Security team that the HIDS logs into Splunk.

4. Actor pushes code, tries to bypass the pull request and tries to merge into master. Splunk detects no Jira Ticket is tagged. Locks out the account.

5. Security + Development team investigate the situation and triage the situation.

PLAN

$$(P+L)(A+N)$$

$$PA + PN + LA + LN$$

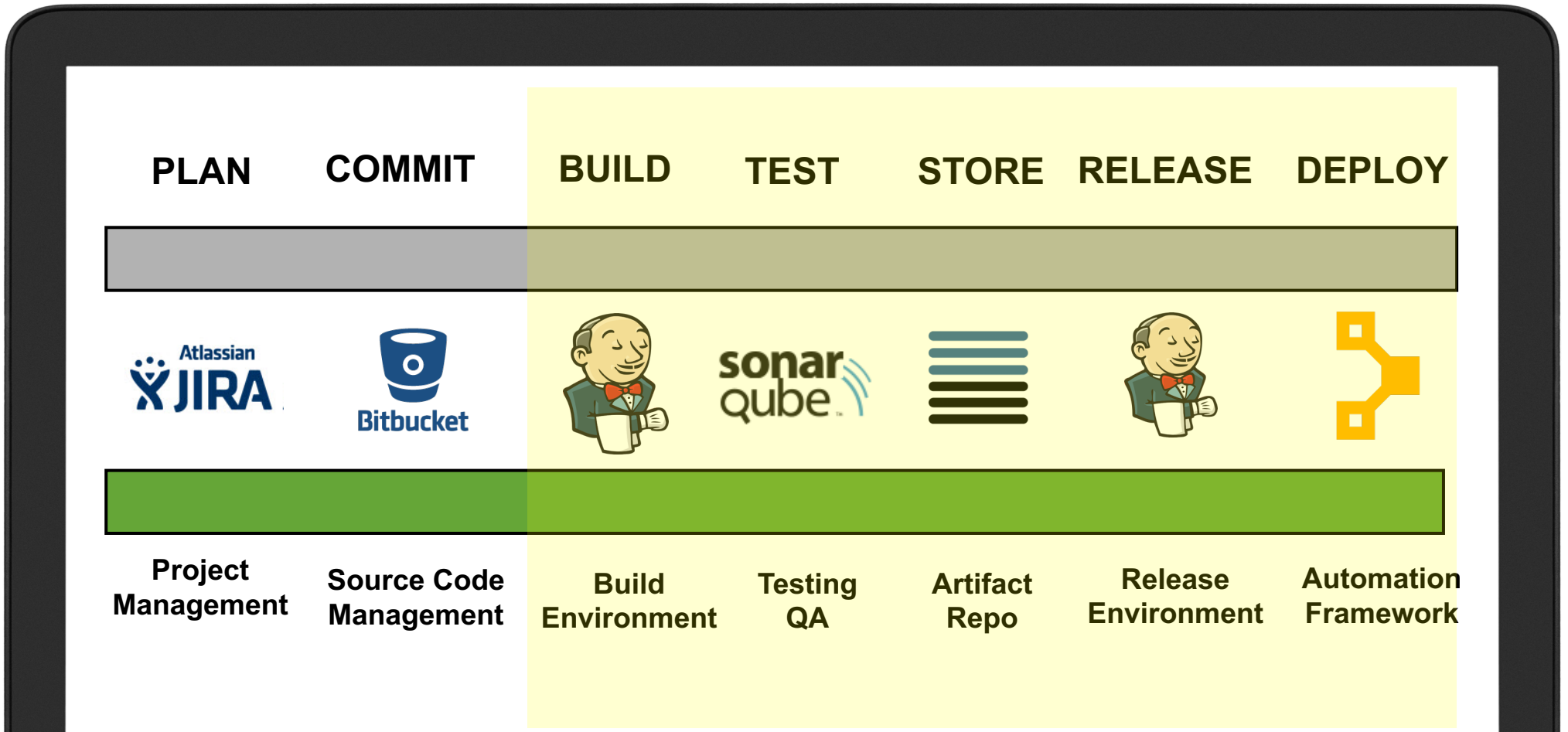
Build Pipeline

Capturing Security relevant information within the Build Pipeline

splunk> .conf2017

Build Pipeline

Plan, Code, Test, Store and Release



Build, Log, Alert

Actor don't typically have to bypass a build plan but sometimes these systems are wide open and don't get logged.

1. Actor bypasses the Repo requirements, passes code reviews and allow the build environment to pull, build, and release Artifacts.

2. Release Engineering team will then blindly push out the binary and stage it for the environment.

3. Actor patiently waits for the moment to strike and code gets embedded into other builds and soon spreads like a a virus.

4. Actor strikes and take over the system by crypto locking all the files, databases, and post a hostage message for the internal employees.

5. Company has to take down the system and deal with a PR mess.

Julianne Helinek

@lookitsjulianne

Pepsi: We're having the worst PR nightmare ever.

United: Hold my beer.

Build, Log, Alert

Actor don't typically have to bypass a build plan but sometimes these systems are wide open and don't get logged.

1. Actor attempts to bypass the security checks and development process but the system is able to log and react to the situation.
2. Actor is locked out of the system.
3. Team investigates and nothing is leaked outside the company environment.
4. Actor Fails in the Final Act.

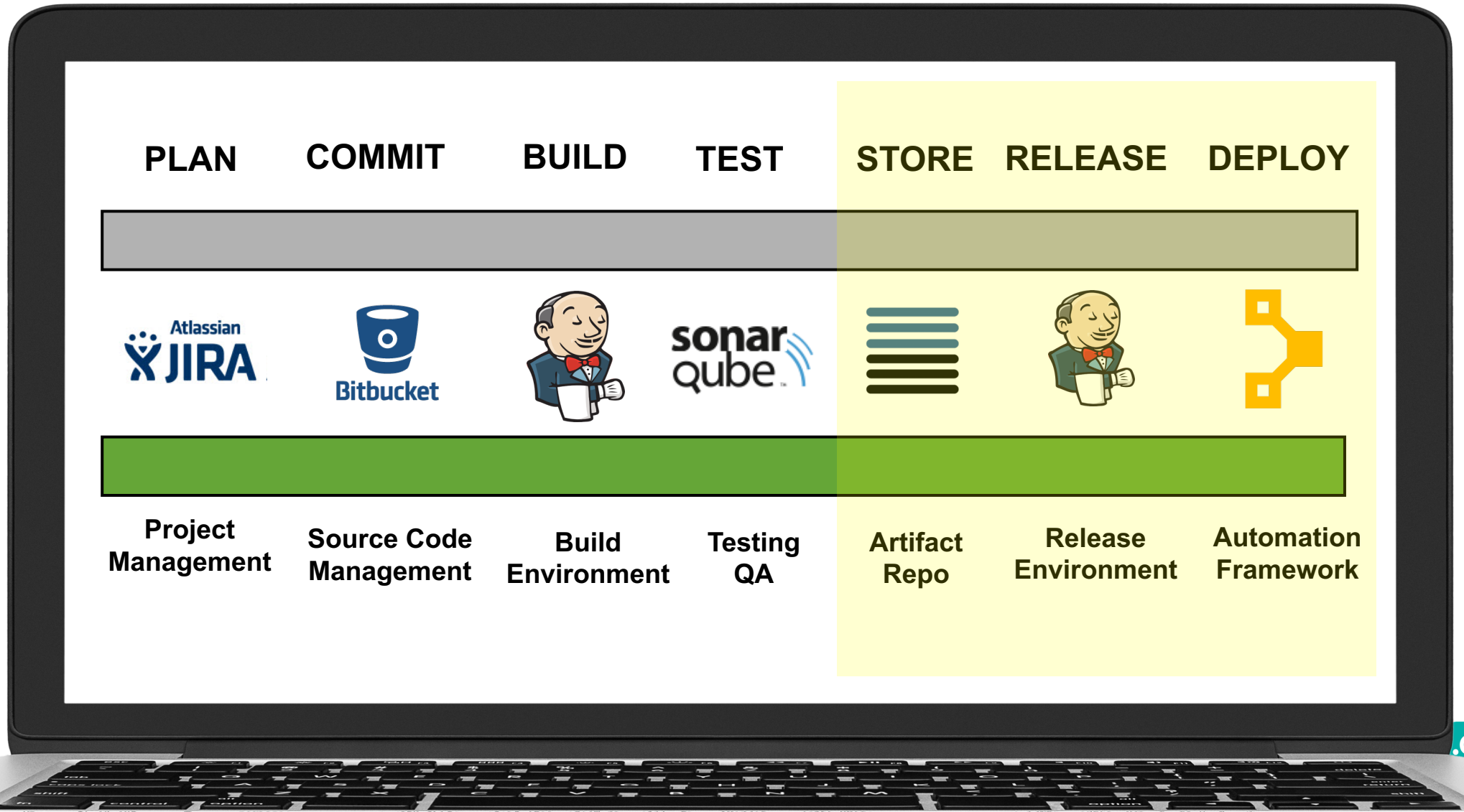
Release Pipeline

Releasing into the Secure Environment

splunk> .conf2017

Release Pipeline

Store, Release, Deploy with Secure Velocity



Release with Confidence

A actor will take all possible open doors to exploit your lack of logging. Don't be that guy.

1. Actor bypass checks in the code review and forces a massive change within the application framework.
2. The build environment creates the artifact with the exploit code and increases the size of the package by 20%.
3. Release Engineering does not notice the abnormal increase of changes and pushes into the environment.
4. Customer has a breach situation and customer data is leaked. PR Mess Begins.


DON'T BE THAT GUY
NOBODY LIKES THAT GUY

Release with Confidence

A actor will take all possible open doors to exploit your lack of logging. Don't be that guy.

1. Actor bypasses the code review process. Pushes changes into the master code branch and forces the change through the build system that creates an artifact.

2. Release Engineering group report notices that package grew by 20% and stops the release of the application. Splunk locks out the account from doing anymore builds or code changes.

3. Developers checks to see what happens, notices the code changes, reverts back the changes and put the code in future release process.

4. Developers do a root case analysis and find the break in the code review process and make the proper adjustments.

Repeat and Learn

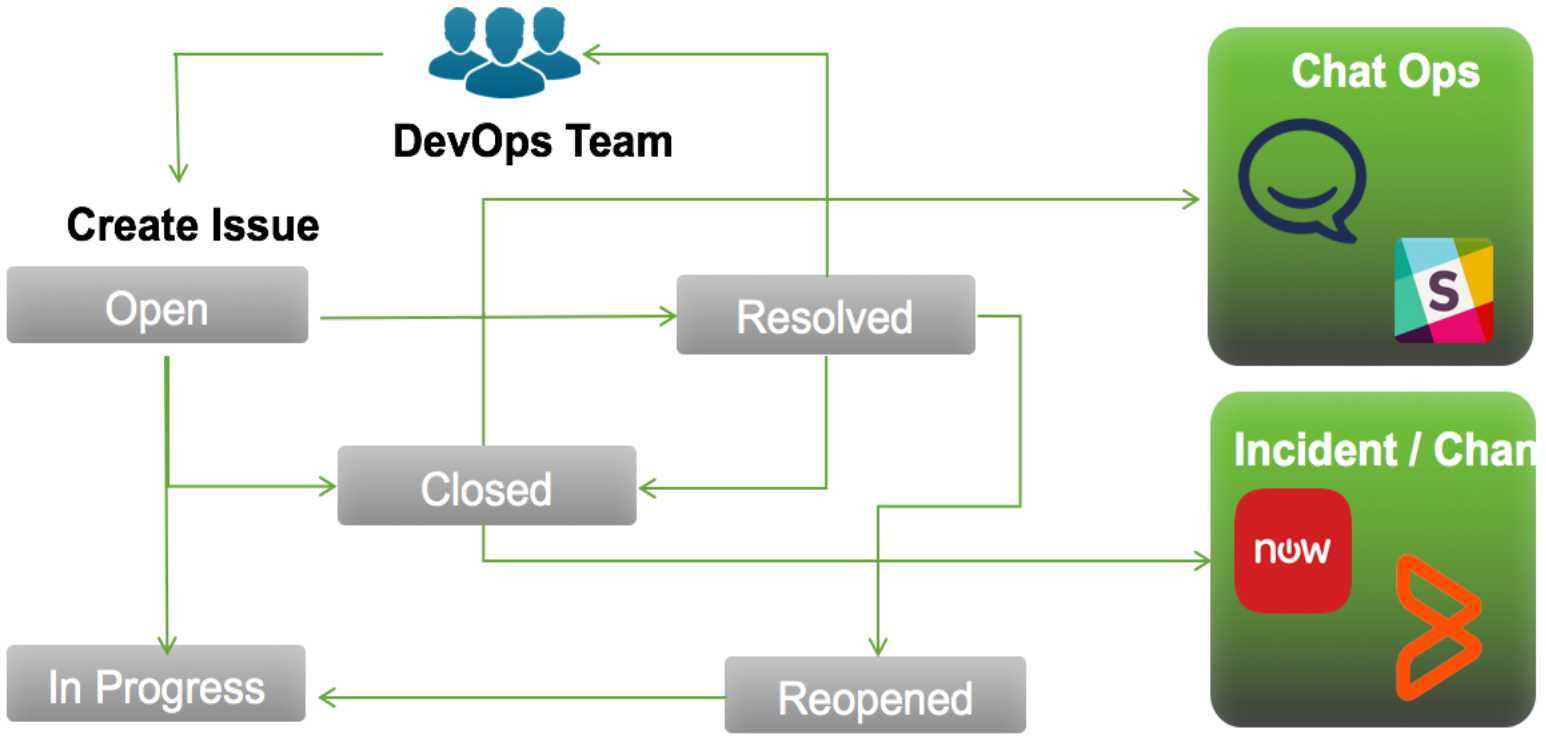
Repeating the Process and Learning from your Data

splunk> .conf2017

Fail… Succeed… Fail… Learn

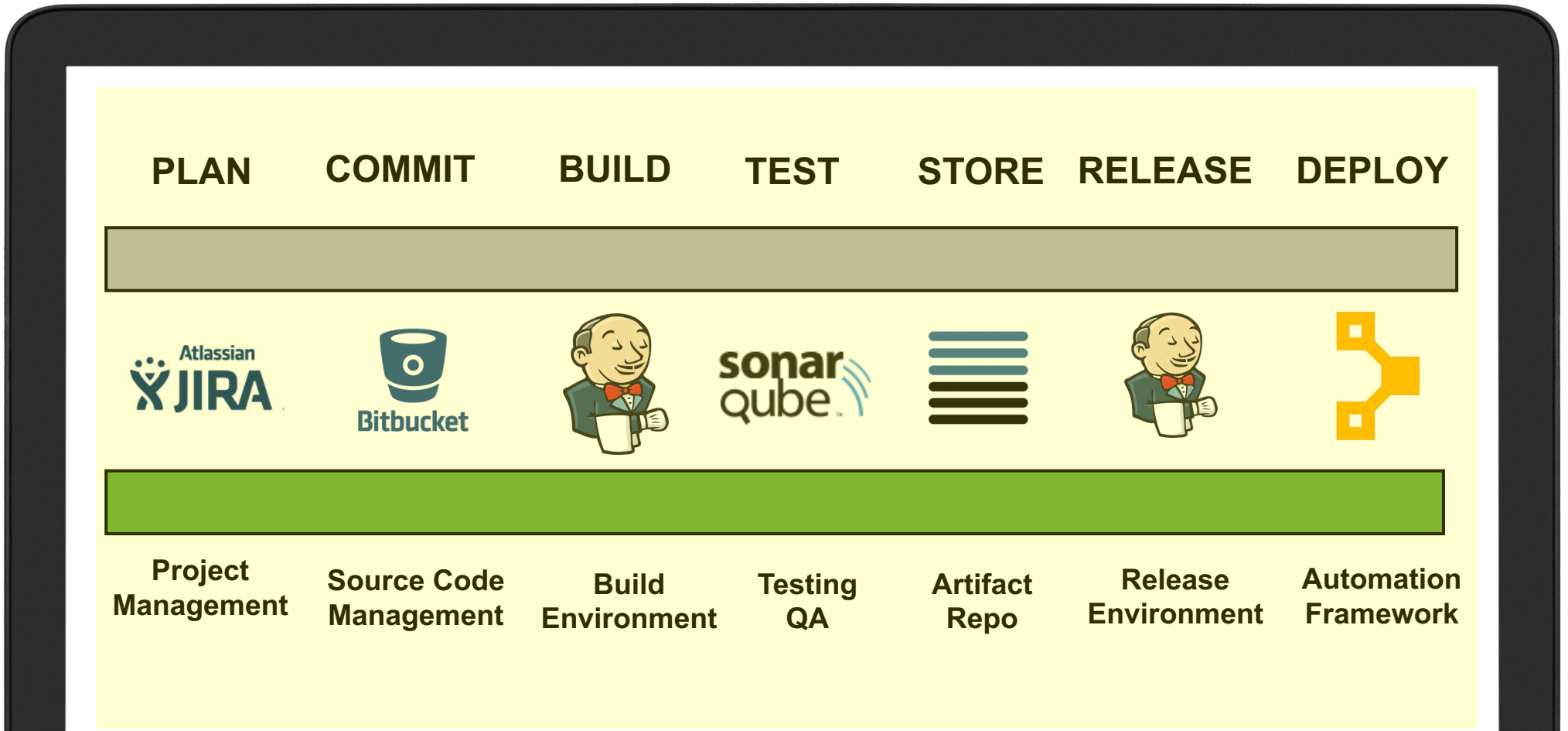
You can only learn from your own mistakes but sometimes it takes a little information to go the right direction.

Plan and Respond to Changes



DevOps Pipeline with Repeat and Learn

By learning from your mistakes you can **Automate** your safe guards.



Thank You

Don't forget to rate this session in the .conf2017 mobile app